Performance Optimization with PerfExpert and MACPO

Jim Browne, Ashay Rane and Leo Fialho

ICS 2013

TACC

THE UNIVERSITY OF TEXAS AT AUSTIN

Agenda

Agenda

Agenda

In the morning:

09:00 Introduction and motivation *[Jim]*

09:20 What PerfExpert can provide to you? *[Leo]*

09:30 Demo *[Leo]*

09:45 How PerfExpert does that? (opening Pandora's box) *[Leo]*

10:15 Extending PerfExpert *[Leo]*

10:30 (Coffee?) break *[everyone, including you]*

10:45 Hands on tutorial *[all the team]*

11:45 Morning closure *[all the team]*

Agenda

Agenda

In the afternoon:

01:30 What MACPO can provide to you? *[Ashay]*

02:00 Demo *[Ashay, Jim]*

02:30 How MACPO does that? *[Ashay]*

03:15 (Coffee?) break *[everyone, including you]*

03:30 Hands on tutorial *[Ashay]*

04:00 Selecting code segments to run on GPUs/accelerators *[Jim]*

04:30 Enhancing PerfExpert with MACPO analysis *[all the team]*

04:45 Afternoon closure and future work *[all the team]*

Overview: why PerfExpert?

Overview: why PerfExpert?

Problem: HPC systems operate far below peak

Overview: why PerfExpert?

Problem: HPC systems operate far below peak

- Chip/node architectural complexity is growing rapidly

Overview: why PerfExpert?

Problem: HPC systems operate far below peak

- Chip/node architectural complexity is growing rapidly
- Performance optimization for these chips requires deep knowledge of architectures, code patterns, compilers, etc.

Performance optimization tools

Overview: why PerfExpert?

Problem: HPC systems operate far below peak

- Chip/node architectural complexity is growing rapidly
- Performance optimization for these chips requires deep knowledge of architectures, code patterns, compilers, etc.

Performance optimization tools

- Powerful in the hands of experts

Overview: why PerfExpert?

Problem: HPC systems operate far below peak

- Chip/node architectural complexity is growing rapidly
- Performance optimization for these chips requires deep knowledge of architectures, code patterns, compilers, etc.

Performance optimization tools

- Powerful in the hands of experts
- Require detailed performance and system expertise

Overview: why PerfExpert?

Problem: HPC systems operate far below peak

- Chip/node architectural complexity is growing rapidly
- Performance optimization for these chips requires deep knowledge of architectures, code patterns, compilers, etc.

Performance optimization tools

- Powerful in the hands of experts
- Require detailed performance and system expertise
- HPC application developers are domain experts, not computer gurus

Overview: why PerfExpert?

Problem: HPC systems operate far below peak

- Chip/node architectural complexity is growing rapidly
- Performance optimization for these chips requires deep knowledge of architectures, code patterns, compilers, etc.

Performance optimization tools

- Powerful in the hands of experts
- Require detailed performance and system expertise
- HPC application developers are domain experts, not computer gurus

Result: Many HPC programmers do not use these tools

(seriously)

Goal for PerfExpert: democratize optimization!

Goal for PerfExpert: democratize optimization!

Subgoals:

Goal for PerfExpert: democratize optimization!

Subgoals:

- Make use of the tool as simple as possible

Goal for PerfExpert: democratize optimization!

Subgoals:

- Make use of the tool as simple as possible
- Start with only chip/node level optimization

Goal for PerfExpert: democratize optimization!

Subgoals:

- Make use of the tool as simple as possible
- Start with only chip/node level optimization
- Make it adaptable across multiple architectures

Goal for PerfExpert: democratize optimization!

Subgoals:

- Make use of the tool as simple as possible
- Start with only chip/node level optimization
- Make it adaptable across multiple architectures
- Design for extension to communication and I/O performance

Goal for PerfExpert: democratize optimization!

Subgoals:

- Make use of the tool as simple as possible
- Start with only chip/node level optimization
- Make it adaptable across multiple architectures
- Design for extension to communication and I/O performance

How to accomplish?

Goal for PerfExpert: democratize optimization!

Subgoals:

- Make use of the tool as simple as possible
- Start with only chip/node level optimization
- Make it adaptable across multiple architectures
- Design for extension to communication and I/O performance

How to accomplish?

- Formulate the performance optimization task as a workflow of subtasks

Goal for PerfExpert: democratize optimization!

Subgoals:

- Make use of the tool as simple as possible
- Start with only chip/node level optimization
- Make it adaptable across multiple architectures
- Design for extension to communication and I/O performance

How to accomplish?

- Formulate the performance optimization task as a workflow of subtasks
- Leverage the state-of-the-art: Build on the best available tools for the subtasks to minimize the effort and cost of development

Goal for PerfExpert: democratize optimization!

Subgoals:

- Make use of the tool as simple as possible
- Start with only chip/node level optimization
- Make it adaptable across multiple architectures
- Design for extension to communication and I/O performance

How to accomplish?

- Formulate the performance optimization task as a workflow of subtasks
- Leverage the state-of-the-art: Build on the best available tools for the subtasks to minimize the effort and cost of development
- Automate the entire workflow

Goal for PerfExpert: democratize optimization!

Subgoals:

- Make use of the tool as simple as possible
- Start with only chip/node level optimization
- Make it adaptable across multiple architectures
- Design for extension to communication and I/O performance

How to accomplish?

- Formulate the performance optimization task as a workflow of subtasks
- Leverage the state-of-the-art: Build on the best available tools for the subtasks to minimize the effort and cost of development
- Automate the entire workflow

Introduction

The four stages of automatic performance optimization:

Introduction

The four stages of automatic performance optimization:

- Measurement and attribution (1)

Introduction

The four stages of automatic performance optimization:

- Measurement and attribution (1)
- Analysis, diagnosis and identification of bottlenecks (2)

Introduction

The four stages of automatic performance optimization:

- Measurement and attribution (1)
- Analysis, diagnosis and identification of bottlenecks (2)
- Selection of effective optimizations (3)

Introduction

The four stages of automatic performance optimization:

- Measurement and attribution (1)
- Analysis, diagnosis and identification of bottlenecks (2)
- Selection of effective optimizations (3)
- Implementation of optimizations (4)

Introduction

The four stages of automatic performance optimization:

- Measurement and attribution (1)
- Analysis, diagnosis and identification of bottlenecks (2)
- Selection of effective optimizations (3)
- Implementation of optimizations (4)

Use of State-of-the-Art:

Introduction

The four stages of automatic performance optimization:

- Measurement and attribution (1)
- Analysis, diagnosis and identification of bottlenecks (2)
- Selection of effective optimizations (3)
- Implementation of optimizations (4)

Use of State-of-the-Art:

- HPCToolkit, **MACPO** based on ROSE (1)

Introduction

The four stages of automatic performance optimization:

- Measurement and attribution (1)
- Analysis, diagnosis and identification of bottlenecks (2)
- Selection of effective optimizations (3)
- Implementation of optimizations (4)

Use of State-of-the-Art:

- HPCToolkit, **MACPO** based on ROSE (1)
- **PerfExpert Team** (2 and 3)

Introduction

The four stages of automatic performance optimization:

- Measurement and attribution (1)
- Analysis, diagnosis and identification of bottlenecks (2)
- Selection of effective optimizations (3)
- Implementation of optimizations (4)

Use of State-of-the-Art:

- HPCToolkit, **MACPO** based on ROSE (1)
- **PerfExpert Team** (2 and 3)
- **PerfExpert Team** based on ROSE, PIPS, Bison and Flex (4)

Introduction

The four stages of automatic performance optimization:

- Measurement and attribution (1)
- Analysis, diagnosis and identification of bottlenecks (2)
- Selection of effective optimizations (3)
- Implementation of optimizations (4)

Use of State-of-the-Art:

- HPCToolkit, **MACPO** based on ROSE (1)
- **PerfExpert Team** (2 and 3)
- **PerfExpert Team** based on ROSE, PIPS, Bison and Flex (4)

Introduction

Uniqueness of PerfExpert:

Introduction

Uniqueness of PerfExpert:

- Nearly complete optimization first three stages of optimization for chip/node level

Introduction

Uniqueness of PerfExpert:

- Nearly complete optimization first three stages of optimization for chip/node level
- Framework for implementing optimizations is complete and several optimizations are completed

Introduction

Uniqueness of PerfExpert:

- Nearly complete optimization first three stages of optimization for chip/node level
- Framework for implementing optimizations is complete and several optimizations are completed
- Integrates code segment focused and data structure based measurements (**MACPO**)

Introduction

Uniqueness of PerfExpert:

- Nearly complete optimization first three stages of optimization for chip/node level
- Framework for implementing optimizations is complete and several optimizations are completed
- Integrates code segment focused and data structure based measurements (**MACPO**)
- Workflow will apply to communication and I/O optimization as well

Introduction

Uniqueness of PerfExpert:

- Nearly complete optimization first three stages of optimization for chip/node level
- Framework for implementing optimizations is complete and several optimizations are completed
- Integrates code segment focused and data structure based measurements (**MACPO**)
- Workflow will apply to communication and I/O optimization as well

Introduction

Unique properties of MACPO:

Introduction

Unique properties of MACPO:

- Multicore resolved traces

Introduction

Unique properties of MACPO:

- Multicore resolved traces
- Code segment local measurement

Introduction

Unique properties of MACPO:

- Multicore resolved traces
- Code segment local measurement
- Data structure specific traces

Introduction

Unique properties of MACPO:

- Multicore resolved traces
- Code segment local measurement
- Data structure specific traces
- Order of magnitude lower overhead of measurement

Introduction

Unique properties of MACPO:

- Multicore resolved traces
- Code segment local measurement
- Data structure specific traces
- Order of magnitude lower overhead of measurement
- More accurate (associative) cache models

Introduction

Unique properties of MACPO:

- Multicore resolved traces
- Code segment local measurement
- Data structure specific traces
- Order of magnitude lower overhead of measurement
- More accurate (associative) cache models
- Strides by data structure and code segment

Introduction

Unique properties of MACPO:

- Multicore resolved traces
- Code segment local measurement
- Data structure specific traces
- Order of magnitude lower overhead of measurement
- More accurate (associative) cache models
- Strides by data structure and code segment
- Architecture “independent” metrics

Introduction

Unique properties of MACPO:

- Multicore resolved traces
- Code segment local measurement
- Data structure specific traces
- Order of magnitude lower overhead of measurement
- More accurate (associative) cache models
- Strides by data structure and code segment
- Architecture “independent” metrics

Agenda

What PerfExpert can provide to you?

Performance report:

What PerfExpert can provide to you?

Performance report:

- Identification of bottlenecks by relevance

What PerfExpert can provide to you?

Performance report:

- Identification of bottlenecks by relevance
- Performance analysis based on performance metrics

What PerfExpert can provide to you?

Performance report:

- Identification of bottlenecks by relevance
- Performance analysis based on performance metrics
- Recommendations for optimization

What PerfExpert can provide to you?

Performance report:

- Identification of bottlenecks by relevance
- Performance analysis based on performance metrics
- Recommendations for optimization

There are three possible outputs:

What PerfExpert can provide to you?

Performance report:

- Identification of bottlenecks by relevance
- Performance analysis based on performance metrics
- Recommendations for optimization

There are three possible outputs:

- Performance report only

What PerfExpert can provide to you?

Performance report:

- Identification of bottlenecks by relevance
- Performance analysis based on performance metrics
- Recommendations for optimization

There are three possible outputs:

- Performance report only
- List of recommendations

What PerfExpert can provide to you?

Performance report:

- Identification of bottlenecks by relevance
- Performance analysis based on performance metrics
- Recommendations for optimization

There are three possible outputs:

- Performance report only
- List of recommendations
- Fully automated code transformation

What PerfExpert can provide to you?

Performance report:

- Identification of bottlenecks by relevance
- Performance analysis based on performance metrics
- Recommendations for optimization

There are three possible outputs:

- Performance report only
- List of recommendations
- Fully automated code transformation

What PerfExpert can provide to you?

Performance report:

What PerfExpert can provide to you?

Performance report:

```
Loop in function compute() at mm.c:8 (99.8% of the total runtime)
================================================================================
ratio to total instrns       % 0.........25..........50.........75........100
   - floating point      :   100 ************************************************
   - data accesses       :    25 ************
* GFLOPS (% max)         :    12 ******
   - packed              :     0 *
   - scalar              :    12 ******
--------------------------------------------------------------------------------
performance assessment     LCPI good......okay......fair......poor......bad....
* overall                :   3.0 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>+
upper bound estimates
* data accesses          :   9.6 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>+
   - L1d hits            :   0.9 >>>>>>>>>>>>>>>>>
   - L2d hits            :   1.8 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
   - L2d misses          :   6.9 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>+
* instruction accesses   :   0.1 >
   - L1i hits            :   0.0 >
   - L2i hits            :   0.0 >
   - L2i misses          :   0.1 >
* data TLB               :   4.6 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>+
* instruction TLB        :   0.0 >
* branch instructions    :   0.1 >>
   - correctly predicted :   0.1 >>
   - mispredicted        :   0.0 >
* floating-point instr   :   5.1 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>+
   - fast FP instr       :   5.1 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>+
   - slow FP instr       :   0.0 >
```

What PerfExpert can provide to you?

List of Recommendations:

What PerfExpert can provide to you?

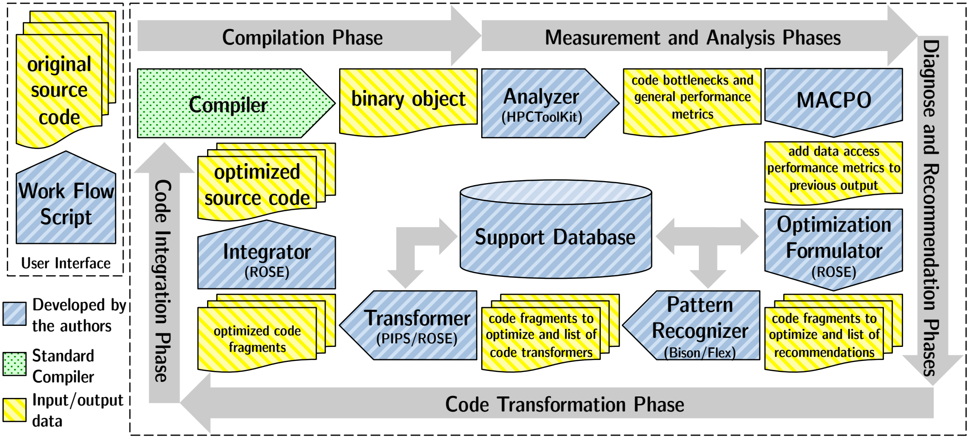
List of Recommendations:

```
#--------------------------------------------------
# Recommendations for mm.c:8
#--------------------------------------------------
#
# This is a possible recommendation for this code segment
#
Recommendation ID: 31
Recommendation Description:  change the order of loops
Recommendation Reason:  this optimization may improve the memory access pattern and make it more
cache and TLB friendly
Pattern Recognizers:  c_loop2 f_loop2
Code example:
loop i {
  loop j {...}
}
=====> loop j {
  loop i {...}
}
```

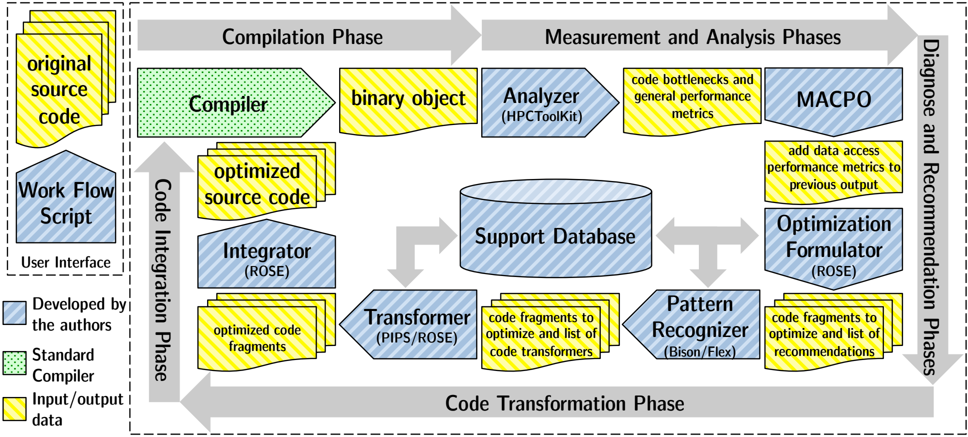
Short Demo

Short demo

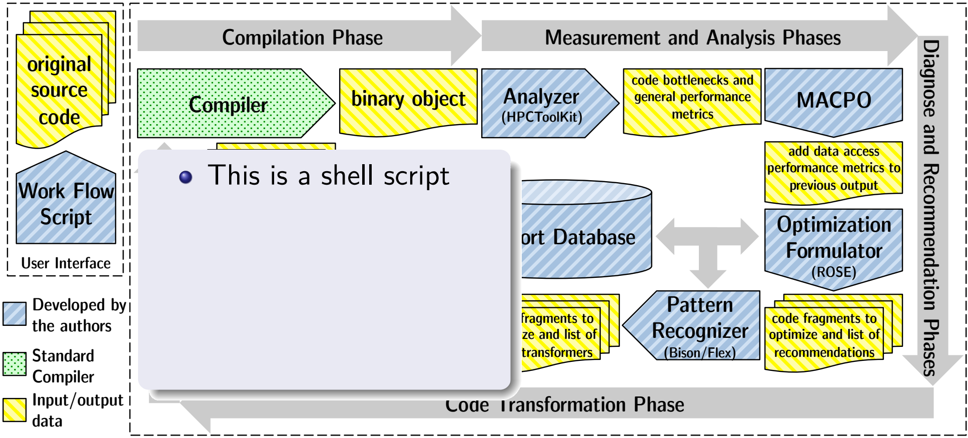
How PerfExpert does that: The Big Picture


original source code
Work Flow Script
User Interface
Developed by the authors
Standard Compiler
Input/output data
Compilation Phase
Measurement and Analysis Phases
Compiler
binary object
Analyzer (HPCToolKit)
code bottlenecks and general performance metrics
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
Diagnose and Recommendation Phases
code fragments to optimize and list of recommendations
Pattern Recognizer (Bison/Flex)
code fragments to optimize and list of code transformers
Transformer (PIPS/ROSE)
Support Database
optimized code fragments
Integrator (ROSE)
optimized source code
Code Integration Phase
Code Transformation Phase

How PerfExpert does that: Work Flow Script

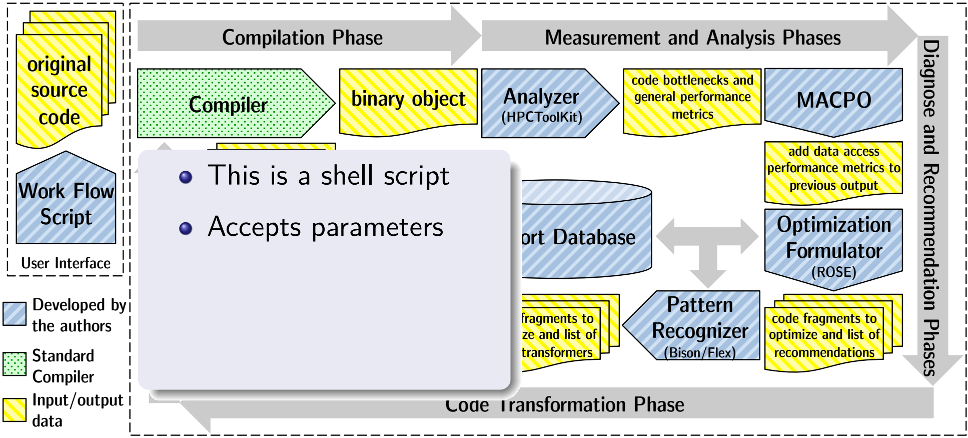

original source code
Work Flow Script
User Interface
Developed by the authors
Standard Compiler
Input/output data
Compilation Phase
Measurement and Analysis Phases
Diagnose and Recommendation Phases
Compiler
binary object
Analyzer (HPCToolKit)
code bottlenecks and general performance metrics
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
code fragments to optimize and list of recommendations
Pattern Recognizer (Bison/Flex)
code fragments to optimize and list of code transformers
Transformer (PIPS/ROSE)
optimized code fragments
Integrator (ROSE)
optimized source code
Support Database
Code Integration Phase
Code Transformation Phase

How PerfExpert does that: Work Flow Script


original source code
Work Flow Script
User Interface
Developed by the authors
Standard Compiler
Input/output data
Compilation Phase
Measurement and Analysis Phases
Compiler
binary object
Analyzer (HPCToolKit)
code bottlenecks and general performance metrics
MACPO
add data access performance metrics to previous output
ort Database
Optimization Formulator (ROSE)
Pattern Recognizer (Bison/Flex)
fragments to ze and list of transformers
code fragments to optimize and list of recommendations
Diagnose and Recommendation Phases
Code Transformation Phase

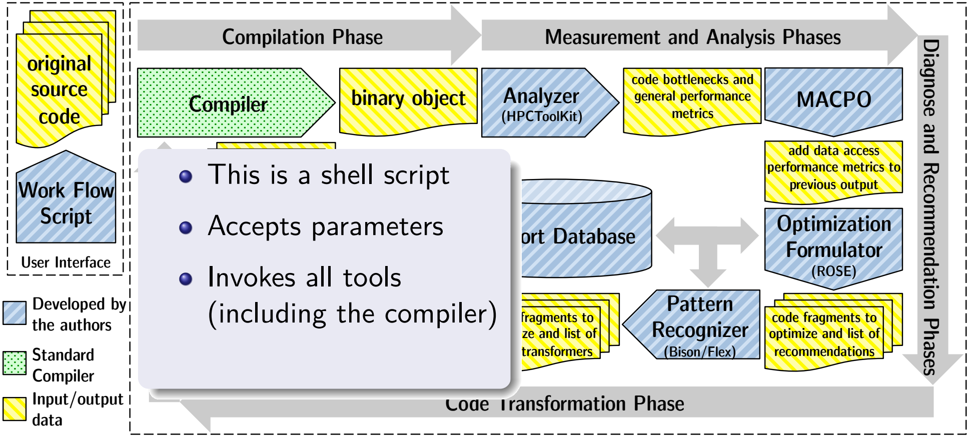
- This is a shell script

How PerfExpert does that: Work Flow Script


original source code
Work Flow Script
User Interface
Developed by the authors
Standard Compiler
Input/output data
Compilation Phase
Measurement and Analysis Phases
Compiler
binary object
Analyzer (HPCToolKit)
code bottlenecks and general performance metrics
MACPO
add data access performance metrics to previous output
ort Database
Optimization Formulator (ROSE)
Pattern Recognizer (Bison/Flex)
fragments to ze and list of transformers
code fragments to optimize and list of recommendations
Diagnose and Recommendation Phases
Code Transformation Phase

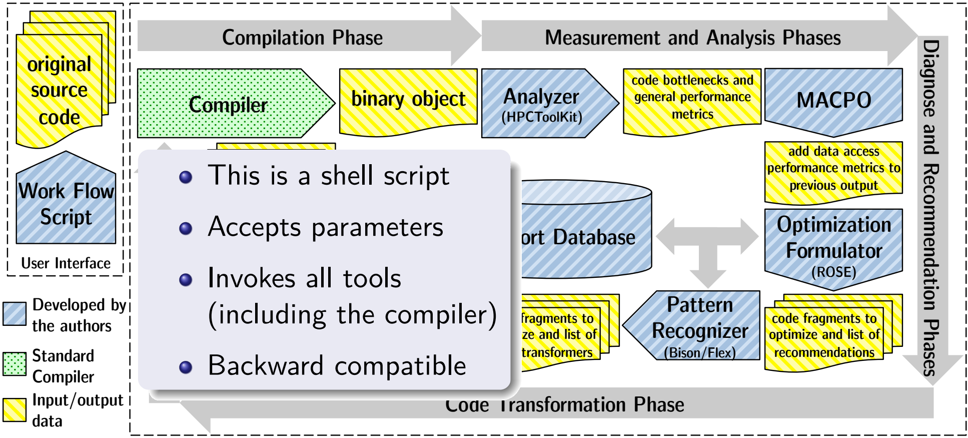
- This is a shell script
- Accepts parameters

How PerfExpert does that: Work Flow Script


original source code
Work Flow Script
User Interface
Developed by the authors
Standard Compiler
Input/output data
Compilation Phase
Measurement and Analysis Phases
Compiler
binary object
Analyzer (HPCToolKit)
code bottlenecks and general performance metrics
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
ort Database
Pattern Recognizer (Bison/Flex)
fragments to ze and list of transformers
code fragments to optimize and list of recommendations
Code Transformation Phase
Diagnose and Recommendation Phases

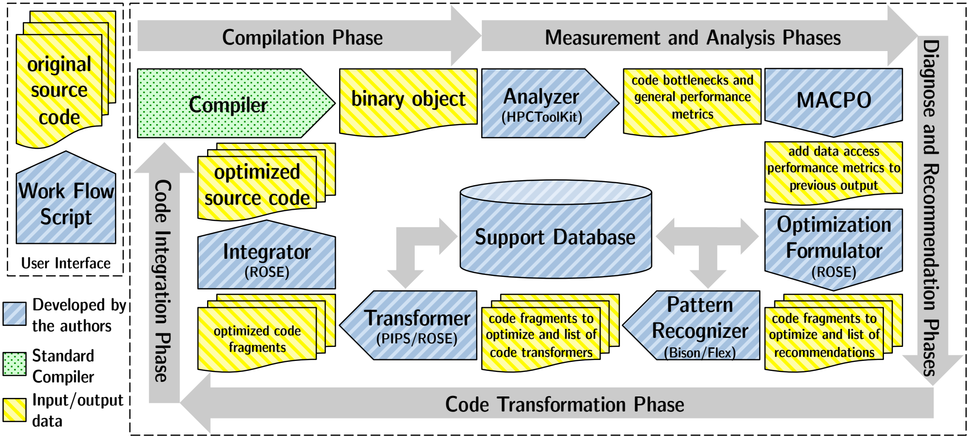
- This is a shell script
- Accepts parameters
- Invokes all tools (including the compiler)

How PerfExpert does that: Work Flow Script

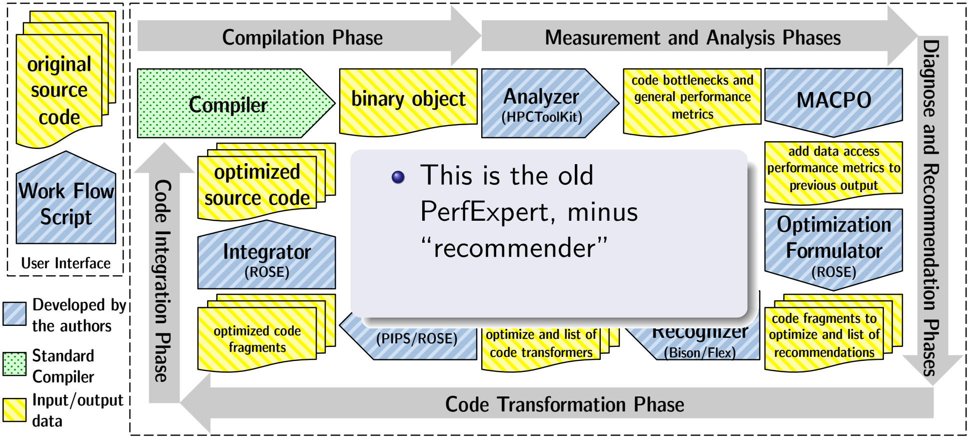

original source code
Work Flow Script
User Interface
Developed by the authors
Standard Compiler
Input/output data
Compilation Phase
Measurement and Analysis Phases
Compiler
binary object
Analyzer (HPCToolKit)
code bottlenecks and general performance metrics
MACPO
add data access performance metrics to previous output
ort Database
Optimization Formulator (ROSE)
fragments to ze and list of transformers
Pattern Recognizer (Bison/Flex)
code fragments to optimize and list of recommendations
Diagnose and Recommendation Phases
Code Transformation Phase

- This is a shell script
- Accepts parameters
- Invokes all tools (including the compiler)
- Backward compatible

How PerfExpert does that: Analyzer

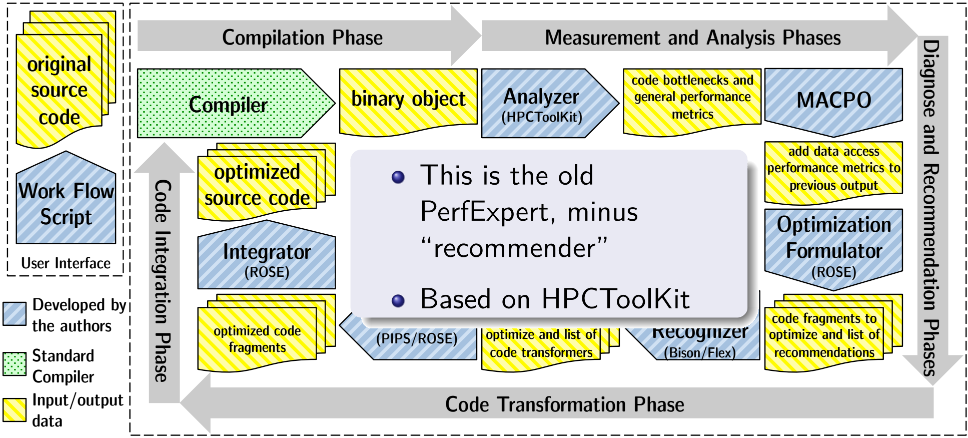

original source code
Work Flow Script
User Interface
Developed by the authors
Standard Compiler
Input/output data
Compilation Phase
Measurement and Analysis Phases
Compiler
binary object
Analyzer (HPCToolKit)
code bottlenecks and general performance metrics
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
code fragments to optimize and list of recommendations
Pattern Recognizer (Bison/Flex)
code fragments to optimize and list of code transformers
Transformer (PIPS/ROSE)
optimized code fragments
Integrator (ROSE)
optimized source code
Support Database
Code Integration Phase
Code Transformation Phase
Diagnose and Recommendation Phases

How PerfExpert does that: Analyzer


original source code
Work Flow Script
User Interface
Developed by the authors
Standard Compiler
Input/output data
Compilation Phase
Measurement and Analysis Phases
Compiler
binary object
Analyzer (HPCToolKit)
code bottlenecks and general performance metrics
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
Diagnose and Recommendation Phases
optimized source code
Integrator (ROSE)
optimized code fragments
Code Integration Phase
(PIPS/ROSE)
optimize and list of code transformers
Recognizer (Bison/Flex)
code fragments to optimize and list of recommendations
Code Transformation Phase

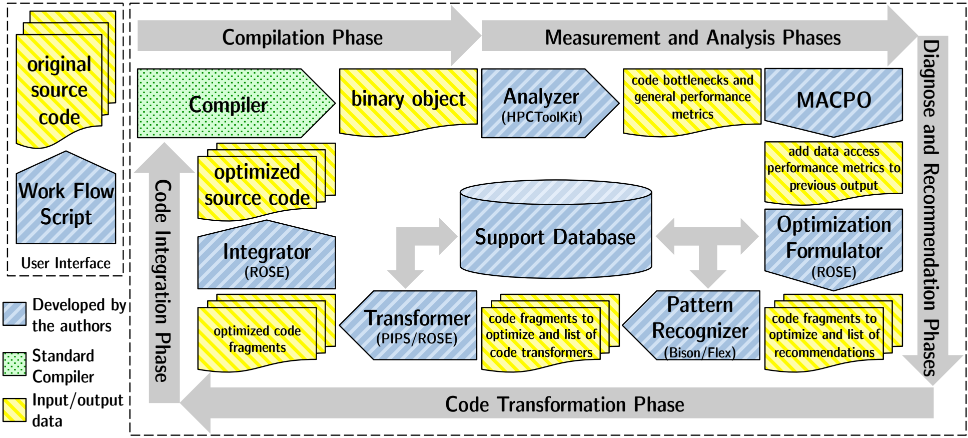
- This is the old PerfExpert, minus “recommender”

How PerfExpert does that: Analyzer

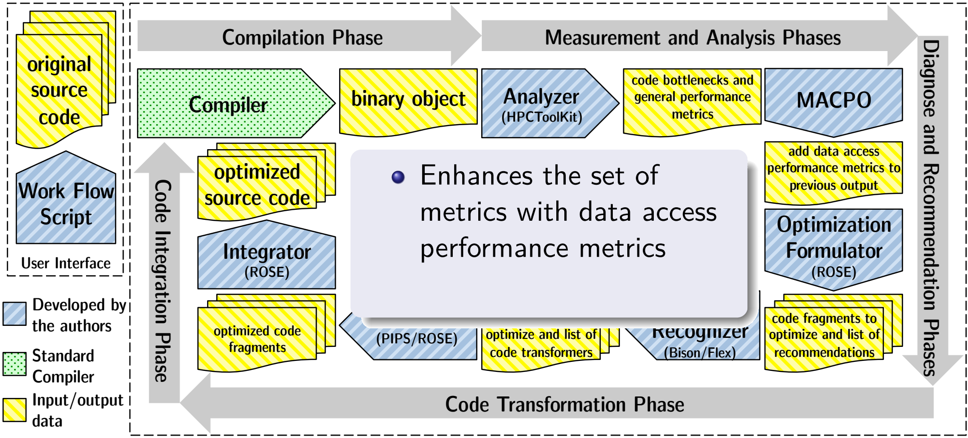

original source code
Work Flow Script
User Interface
Developed by the authors
Standard Compiler
Input/output data
Compilation Phase
Measurement and Analysis Phases
Compiler
binary object
Analyzer (HPCToolKit)
code bottlenecks and general performance metrics
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
code fragments to optimize and list of recommendations
Diagnose and Recommendation Phases
Recognizer (Bison/Flex)
optimize and list of code transformers
(PIPS/ROSE)
optimized code fragments
Integrator (ROSE)
optimized source code
Code Integration Phase
Code Transformation Phase

- This is the old PerfExpert, minus "recommender"
- Based on HPCToolKit

How PerfExpert does that: MACPO

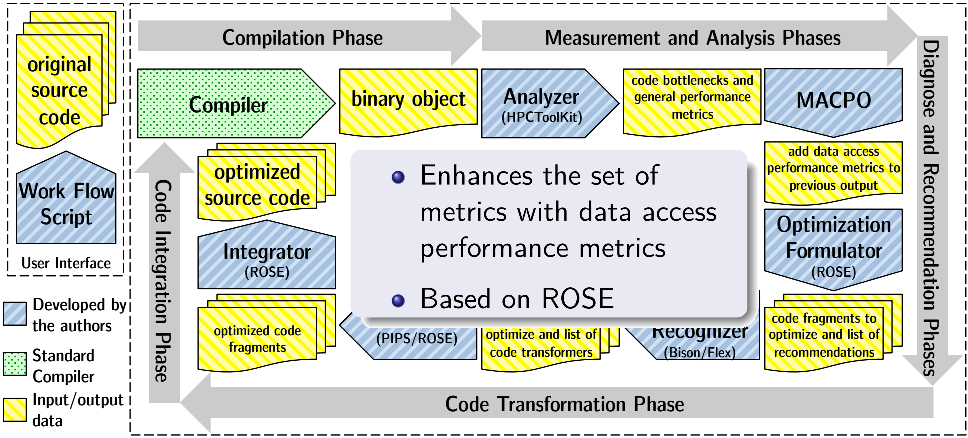

original source code
Work Flow Script
User Interface
Developed by the authors
Standard Compiler
Input/output data
Compilation Phase
Measurement and Analysis Phases
Compiler
binary object
Analyzer (HPCToolKit)
code bottlenecks and general performance metrics
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
Diagnose and Recommendation Phases
optimized source code
Integrator (ROSE)
Support Database
Code Integration Phase
optimized code fragments
Transformer (PIPS/ROSE)
code fragments to optimize and list of code transformers
Pattern Recognizer (Bison/Flex)
code fragments to optimize and list of recommendations
Code Transformation Phase

How PerfExpert does that: MACPO


original source code
Work Flow Script
User Interface
Developed by the authors
Standard Compiler
Input/output data
Compilation Phase
Measurement and Analysis Phases
Compiler
binary object
Analyzer (HPCToolKit)
code bottlenecks and general performance metrics
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
code fragments to optimize and list of recommendations
Diagnose and Recommendation Phases
Code Integration Phase
optimized source code
Integrator (ROSE)
optimized code fragments
(PIPS/ROSE)
optimize and list of code transformers
Recognizer (Bison/Flex)
Code Transformation Phase

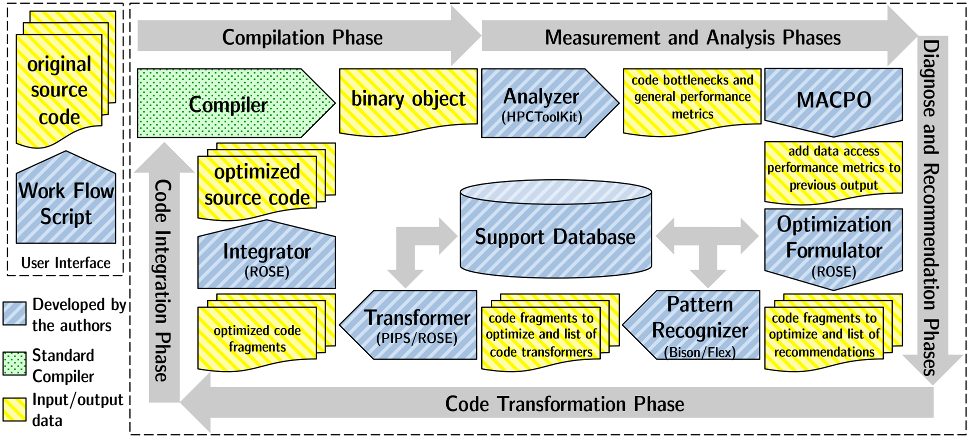
- Enhances the set of metrics with data access performance metrics

How PerfExpert does that: MACPO

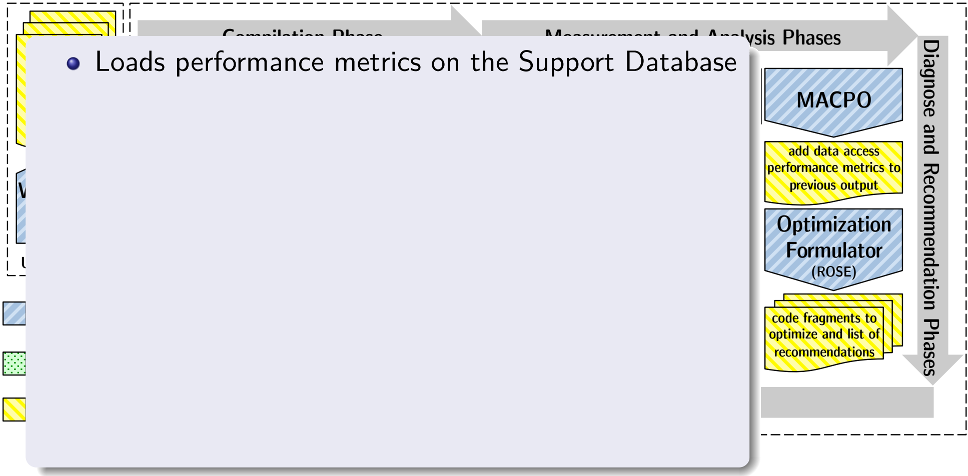

original source code
Work Flow Script
User Interface
Developed by the authors
Standard Compiler
Input/output data
Compilation Phase
Measurement and Analysis Phases
Diagnose and Recommendation Phases
Compiler
binary object
Analyzer (HPCToolKit)
code bottlenecks and general performance metrics
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
code fragments to optimize and list of recommendations
Recognizer (Bison/Flex)
optimize and list of code transformers
(PIPS/ROSE)
optimized code fragments
Integrator (ROSE)
optimized source code
Code Integration Phase
Code Transformation Phase

- Enhances the set of metrics with data access performance metrics
- Based on ROSE

How PerfExpert does that: Optimization Formulator


original source code
Work Flow Script
User Interface
Developed by the authors
Standard Compiler
Input/output data
Compilation Phase
Measurement and Analysis Phases
Compiler
binary object
Analyzer (HPCToolKit)
code bottlenecks and general performance metrics
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
Diagnose and Recommendation Phases
code fragments to optimize and list of recommendations
Pattern Recognizer (Bison/Flex)
code fragments to optimize and list of code transformers
Support Database
Transformer (PIPS/ROSE)
optimized code fragments
Integrator (ROSE)
optimized source code
Code Integration Phase
Code Transformation Phase

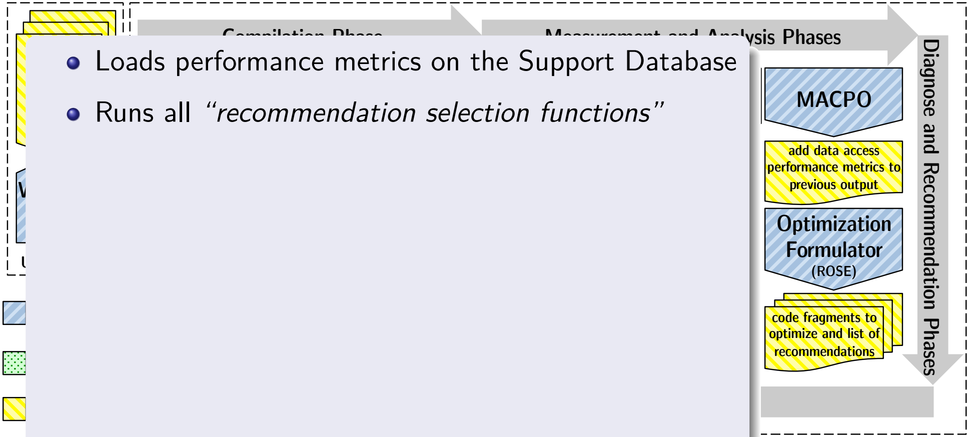
How PerfExpert does that: Optimization Formulator



Compilation Phase
Measurement and Analysis Phases
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
code fragments to optimize and list of recommendations
Diagnose and Recommendation Phases

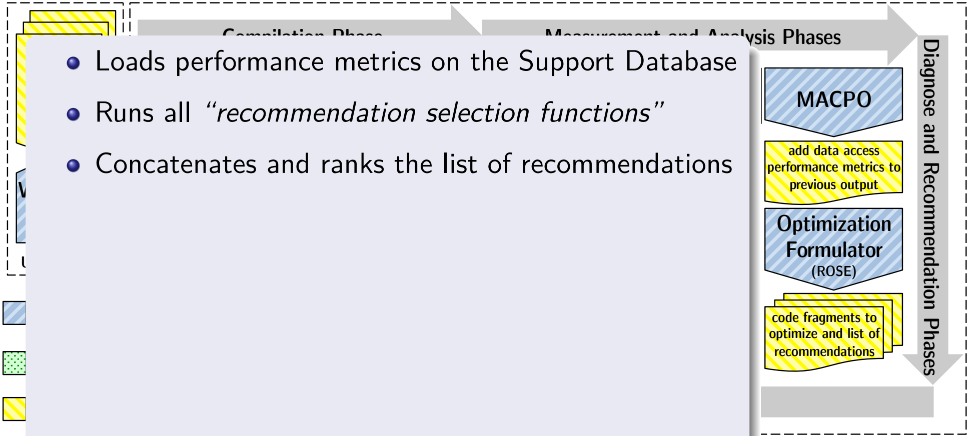
- Loads performance metrics on the Support Database

How PerfExpert does that: Optimization Formulator


Compilation Phase
Measurement and Analysis Phases
Diagnose and Recommendation Phases
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
code fragments to optimize and list of recommendations

- Loads performance metrics on the Support Database
- Runs all *"recommendation selection functions"*

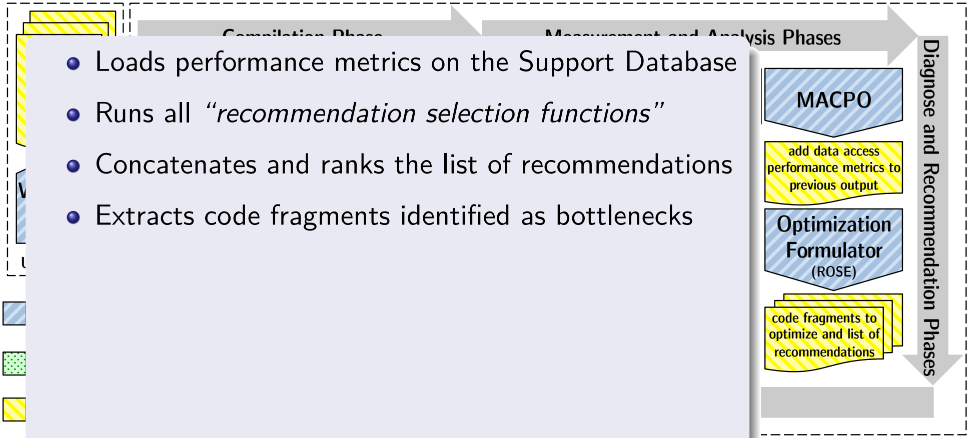
How PerfExpert does that: Optimization Formulator


Compilation Phase
Measurement and Analysis Phases
Diagnose and Recommendation Phases
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
code fragments to optimize and list of recommendations

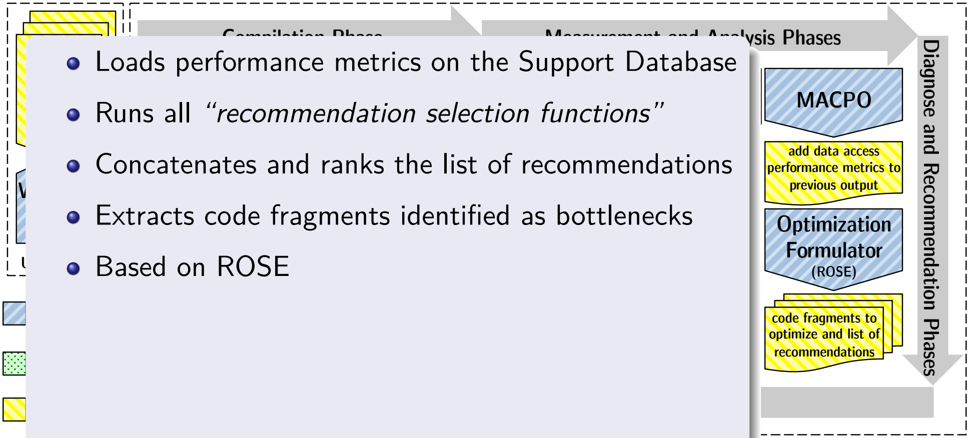
- Loads performance metrics on the Support Database
- Runs all *"recommendation selection functions"*
- Concatenates and ranks the list of recommendations

How PerfExpert does that: Optimization Formulator

- Loads performance metrics on the Support Database
- Runs all *“recommendation selection functions”*
- Concatenates and ranks the list of recommendations
- Extracts code fragments identified as bottlenecks

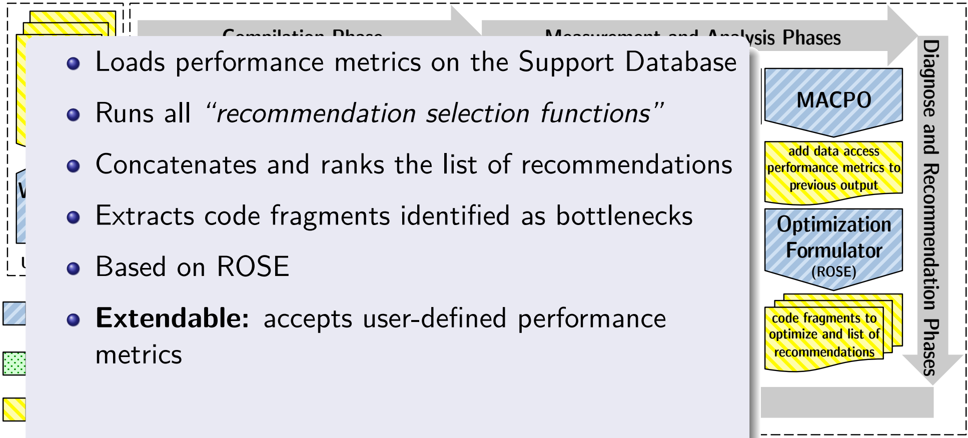

Compilation Phase
Measurement and Analysis Phases
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
code fragments to optimize and list of recommendations
Diagnose and Recommendation Phases

How PerfExpert does that: Optimization Formulator


Compilation Phase
Measurement and Analysis Phases
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
code fragments to optimize and list of recommendations
Diagnose and Recommendation Phases

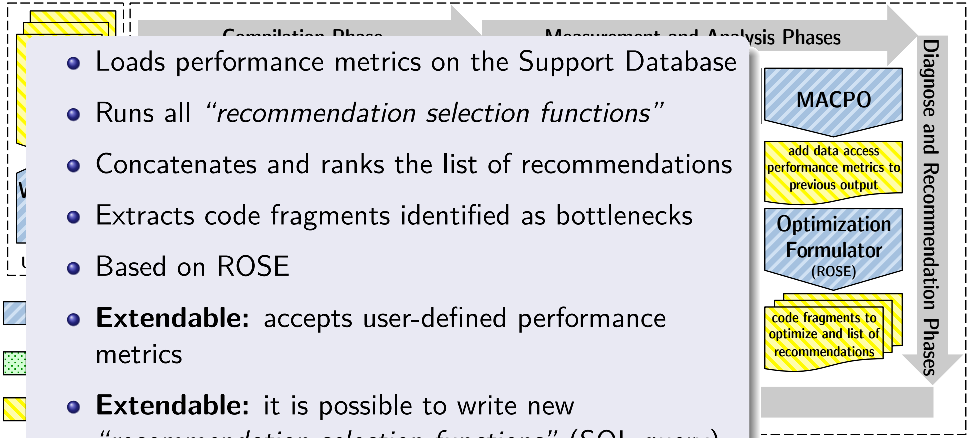
- Loads performance metrics on the Support Database
- Runs all *"recommendation selection functions"*
- Concatenates and ranks the list of recommendations
- Extracts code fragments identified as bottlenecks
- Based on ROSE

How PerfExpert does that: Optimization Formulator


Compilation Phase
Measurement and Analysis Phases
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
code fragments to optimize and list of recommendations
Diagnose and Recommendation Phases

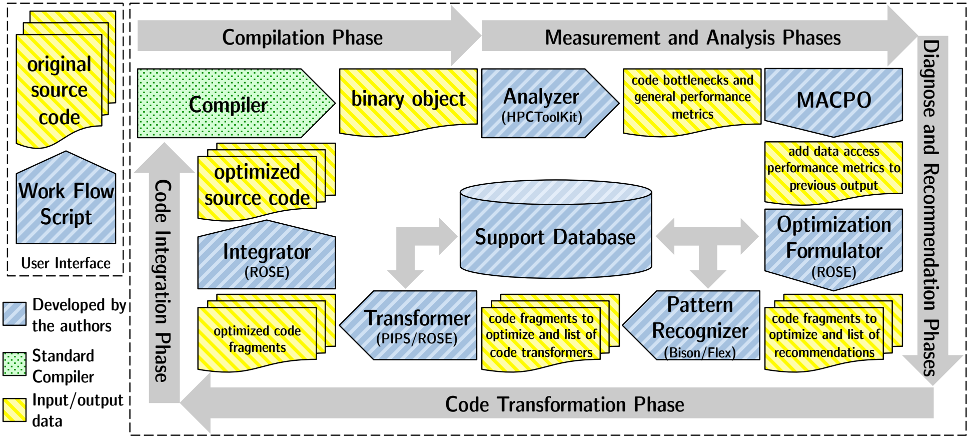
- Loads performance metrics on the Support Database
- Runs all *“recommendation selection functions”*
- Concatenates and ranks the list of recommendations
- Extracts code fragments identified as bottlenecks
- Based on ROSE
- **Extendable:** accepts user-defined performance metrics

How PerfExpert does that: Optimization Formulator


Compilation Phase
Measurement and Analysis Phases
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
code fragments to optimize and list of recommendations
Diagnose and Recommendation Phases

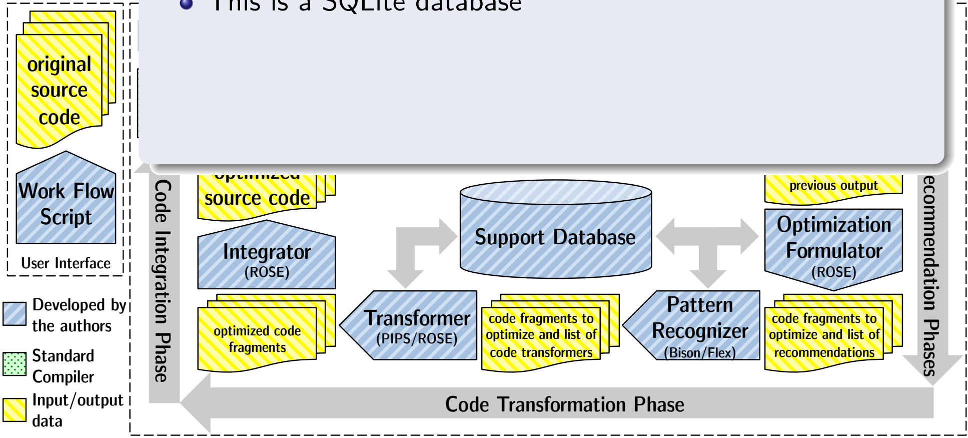
- Loads performance metrics on the Support Database
- Runs all *"recommendation selection functions"*
- Concatenates and ranks the list of recommendations
- Extracts code fragments identified as bottlenecks
- Based on ROSE
- **Extendable:** accepts user-defined performance metrics
- **Extendable:** it is possible to write new *"recommendation selection functions"* (SQL query)

How PerfExpert does that: Support Database


original source code
Work Flow Script
User Interface
Developed by the authors
Standard Compiler
Input/output data
Compilation Phase
Measurement and Analysis Phases
Compiler
binary object
Analyzer (HPCToolKit)
code bottlenecks and general performance metrics
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
Diagnose and Recommendation Phases
optimized source code
Integrator (ROSE)
Support Database
optimized code fragments
Transformer (PIPS/ROSE)
code fragments to optimize and list of code transformers
Pattern Recognizer (Bison/Flex)
code fragments to optimize and list of recommendations
Code Integration Phase
Code Transformation Phase

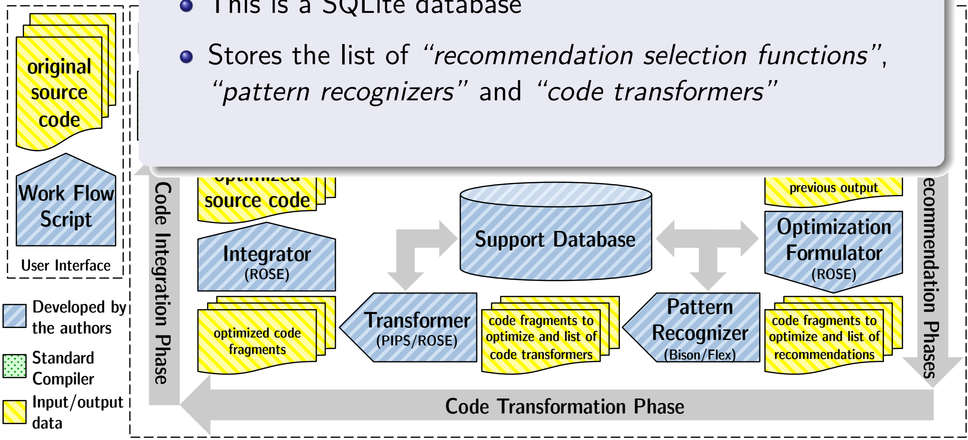
How PerfExpert does that: Support Database

- This is a SQLite database

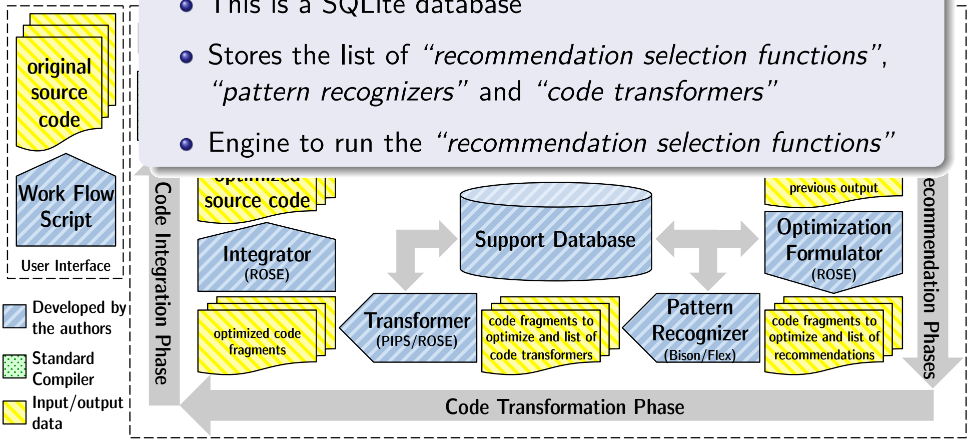


How PerfExpert does that: Support Database

- This is a SQLite database
- Stores the list of *“recommendation selection functions”*, *“pattern recognizers”* and *“code transformers”*

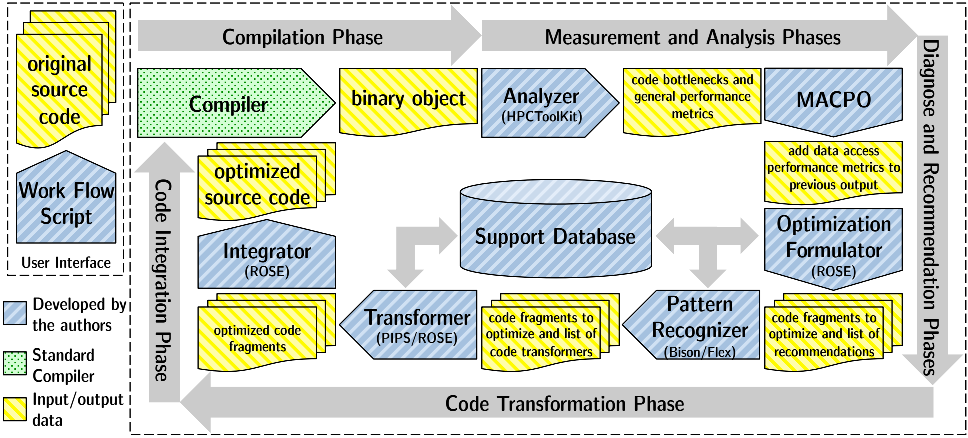


How PerfExpert does that: Support Database



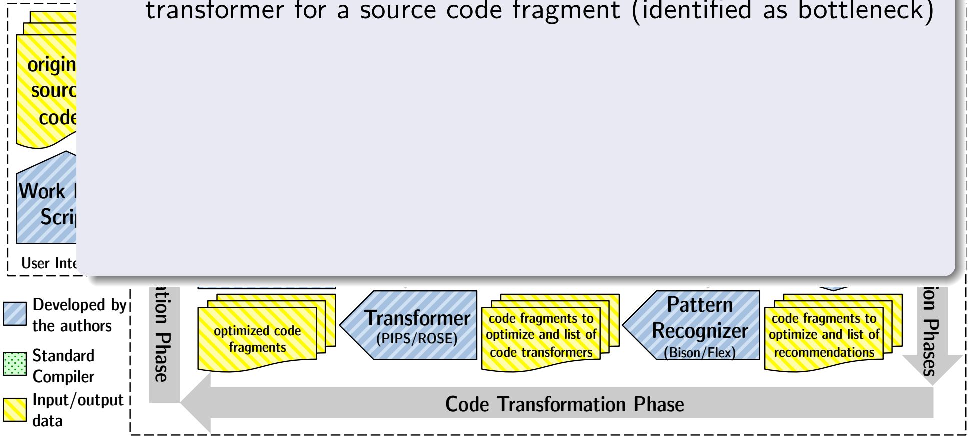
- This is a SQLite database
- Stores the list of *"recommendation selection functions"*, *"pattern recognizers"* and *"code transformers"*
- Engine to run the *"recommendation selection functions"*

How PerfExpert does that: Pattern Recognizer


original source code
Work Flow Script
User Interface
Developed by the authors
Standard Compiler
Input/output data
Compilation Phase
Measurement and Analysis Phases
Compiler
binary object
Analyzer (HPCToolKit)
code bottlenecks and general performance metrics
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
Diagnose and Recommendation Phases
code fragments to optimize and list of recommendations
Pattern Recognizer (Bison/Flex)
code fragments to optimize and list of code transformers
Transformer (PIPS/ROSE)
optimized code fragments
Integrator (ROSE)
optimized source code
Support Database
Code Integration Phase
Code Transformation Phase

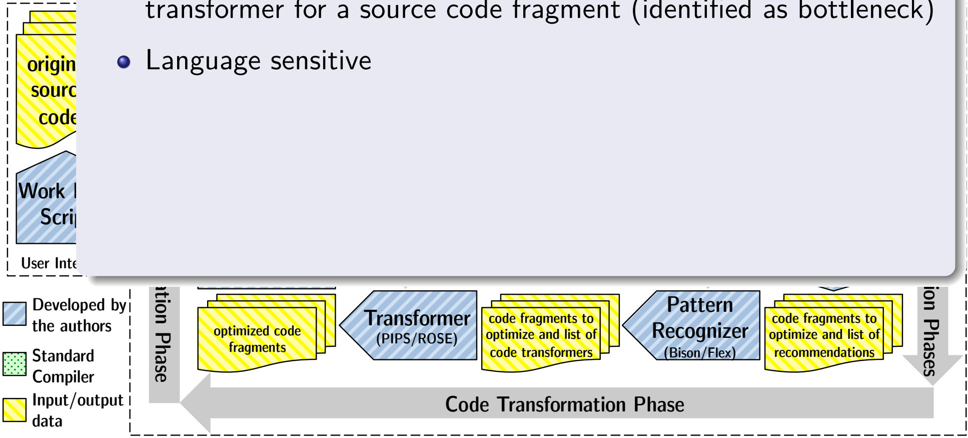
How PerfExpert does that: Pattern Recognizer

- Acts as a "filter" trying to find (match) the right code transformer for a source code fragment (identified as bottleneck)



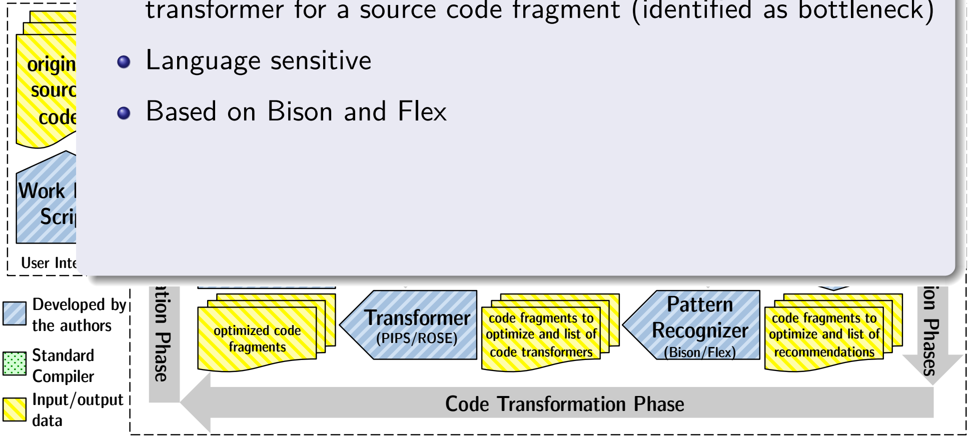
How PerfExpert does that: Pattern Recognizer

- Acts as a “filter” trying to find (match) the right code transformer for a source code fragment (identified as bottleneck)
- Language sensitive



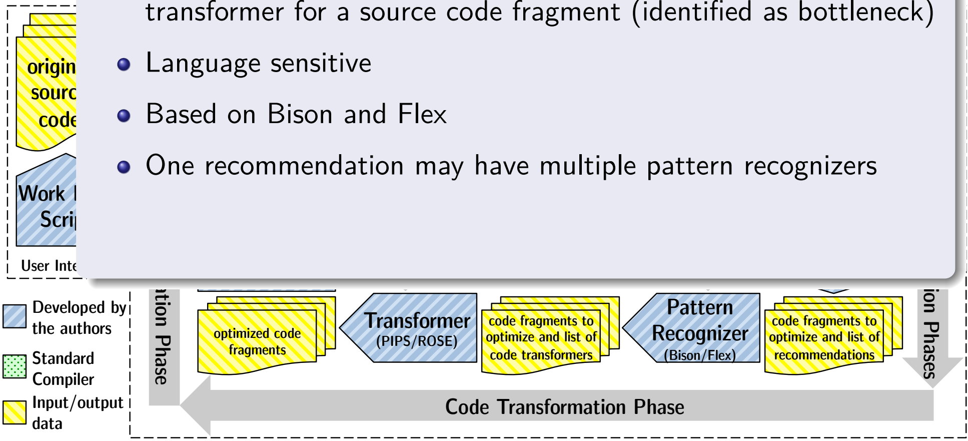
How PerfExpert does that: Pattern Recognizer

- Acts as a "filter" trying to find (match) the right code transformer for a source code fragment (identified as bottleneck)
- Language sensitive
- Based on Bison and Flex



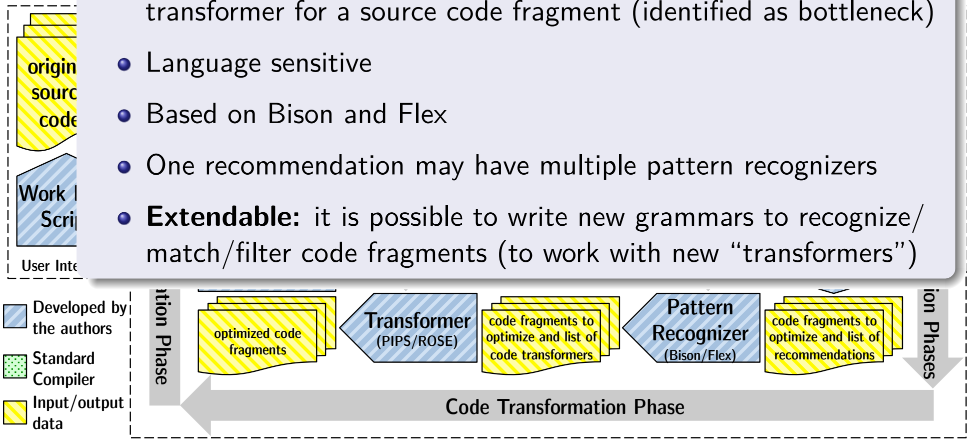
How PerfExpert does that: Pattern Recognizer

- Acts as a “filter” trying to find (match) the right code transformer for a source code fragment (identified as bottleneck)
- Language sensitive
- Based on Bison and Flex
- One recommendation may have multiple pattern recognizers

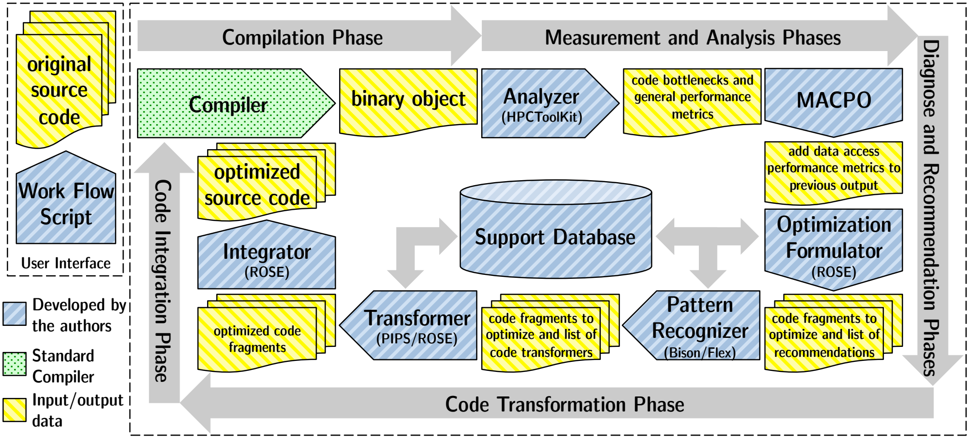


How PerfExpert does that: Pattern Recognizer

- Acts as a “filter” trying to find (match) the right code transformer for a source code fragment (identified as bottleneck)
- Language sensitive
- Based on Bison and Flex
- One recommendation may have multiple pattern recognizers
- **Extendable:** it is possible to write new grammars to recognize/match/filter code fragments (to work with new “transformers”)

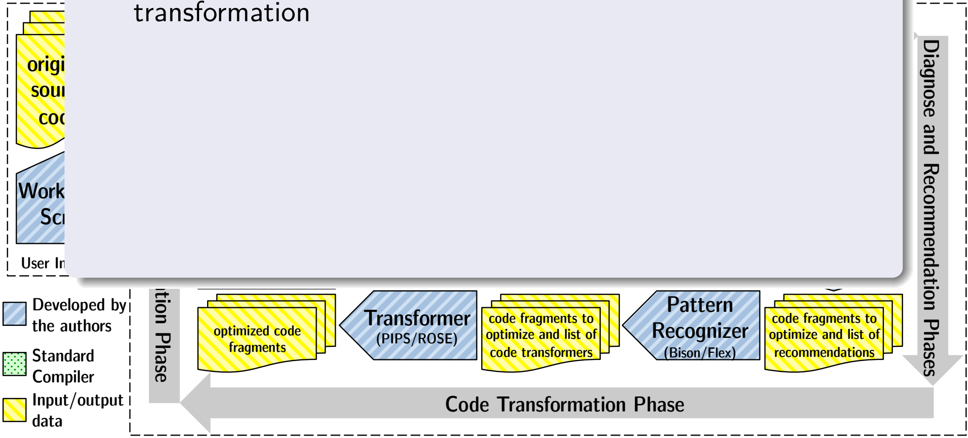


How PerfExpert does that: Transformer


original source code
Work Flow Script
User Interface
Developed by the authors
Standard Compiler
Input/output data
Compilation Phase
Measurement and Analysis Phases
Compiler
binary object
Analyzer (HPCToolKit)
code bottlenecks and general performance metrics
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
Diagnose and Recommendation Phases
code fragments to optimize and list of recommendations
Pattern Recognizer (Bison/Flex)
code fragments to optimize and list of code transformers
Transformer (PIPS/ROSE)
optimized code fragments
Integrator (ROSE)
optimized source code
Support Database
Code Integration Phase
Code Transformation Phase

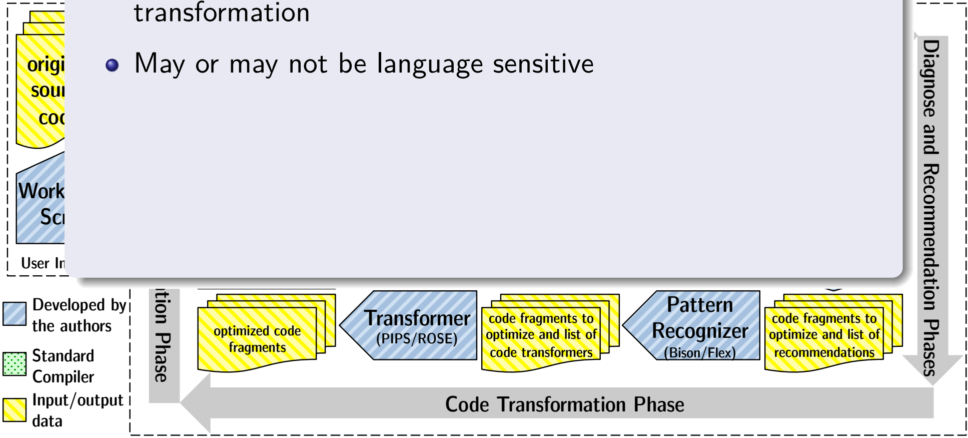
How PerfExpert does that: Transformer

- Implements the recommendation by applying source code transformation



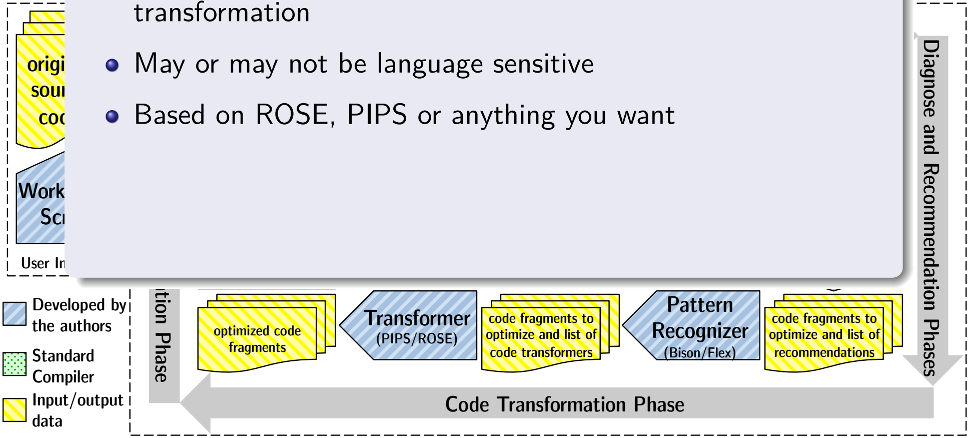
How PerfExpert does that: Transformer

- Implements the recommendation by applying source code transformation
- May or may not be language sensitive



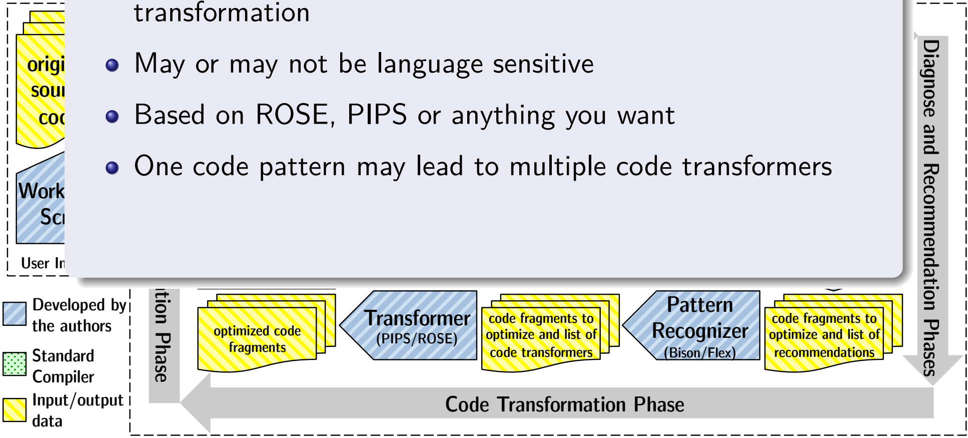
How PerfExpert does that: Transformer

- Implements the recommendation by applying source code transformation
- May or may not be language sensitive
- Based on ROSE, PIPS or anything you want


origi
sour
coo
Work
Scr
User In
Diagnose and Recommendation Phases
Developed by the authors
Standard Compiler
Input/output data
tion Phase
optimized code fragments
Transformer (PIPS/ROSE)
code fragments to optimize and list of code transformers
Pattern Recognizer (Bison/Flex)
code fragments to optimize and list of recommendations
Code Transformation Phase

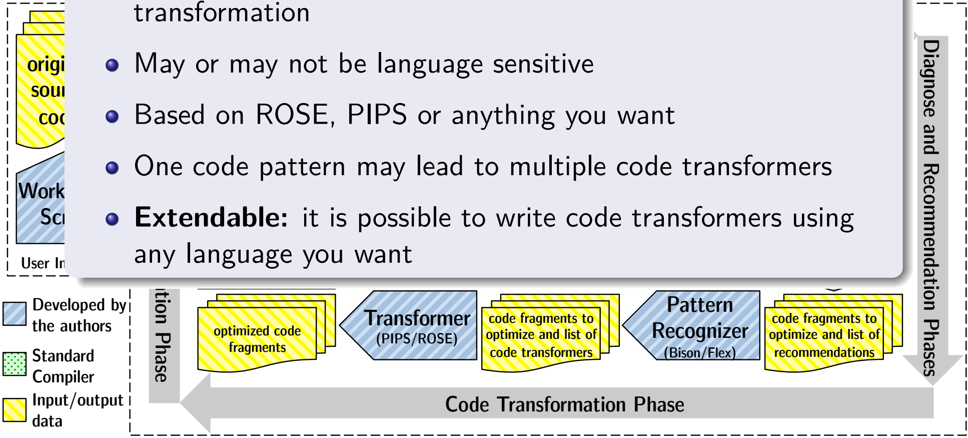
How PerfExpert does that: Transformer

- Implements the recommendation by applying source code transformation
- May or may not be language sensitive
- Based on ROSE, PIPS or anything you want
- One code pattern may lead to multiple code transformers

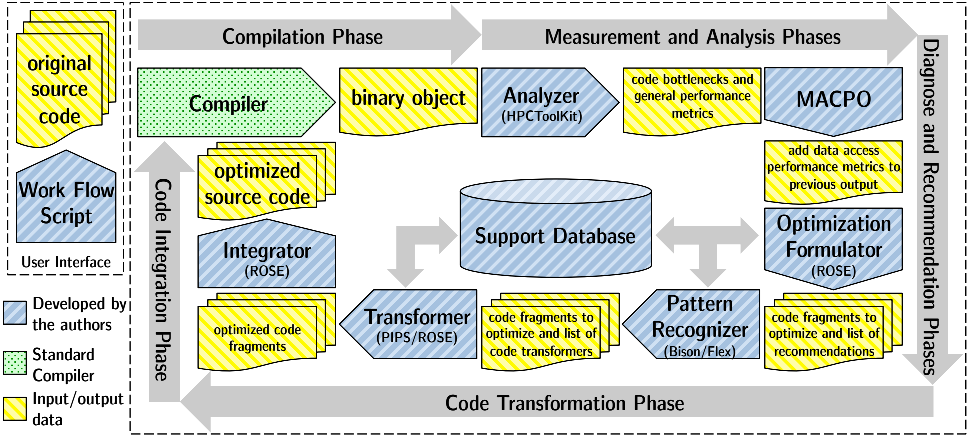


How PerfExpert does that: Transformer

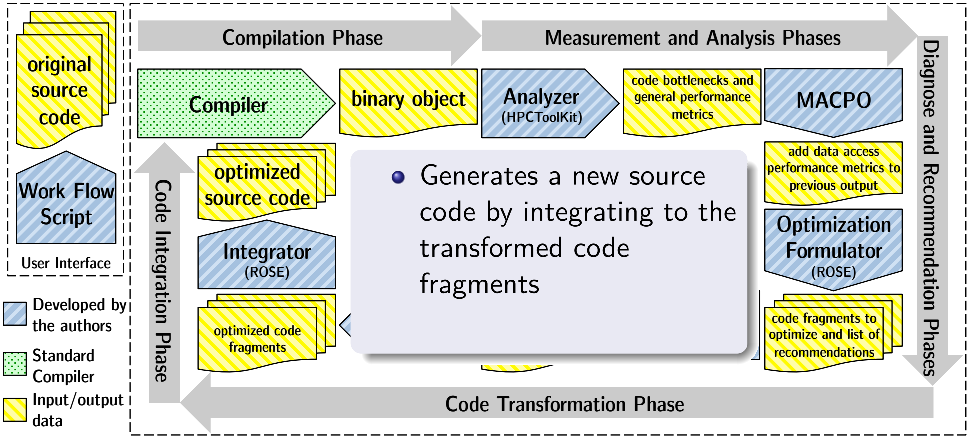
- Implements the recommendation by applying source code transformation
- May or may not be language sensitive
- Based on ROSE, PIPS or anything you want
- One code pattern may lead to multiple code transformers
- **Extendable:** it is possible to write code transformers using any language you want


origi
sou
coc
Work
Sc
User In
Developed by the authors
Standard Compiler
Input/output data
tion Phase
optimized code fragments
Transformer
(PIPS/ROSE)
code fragments to optimize and list of code transformers
Pattern Recognizer
(Bison/Flex)
code fragments to optimize and list of recommendations
Diagnose and Recommendation Phases
Code Transformation Phase

How PerfExpert does that: Integrator

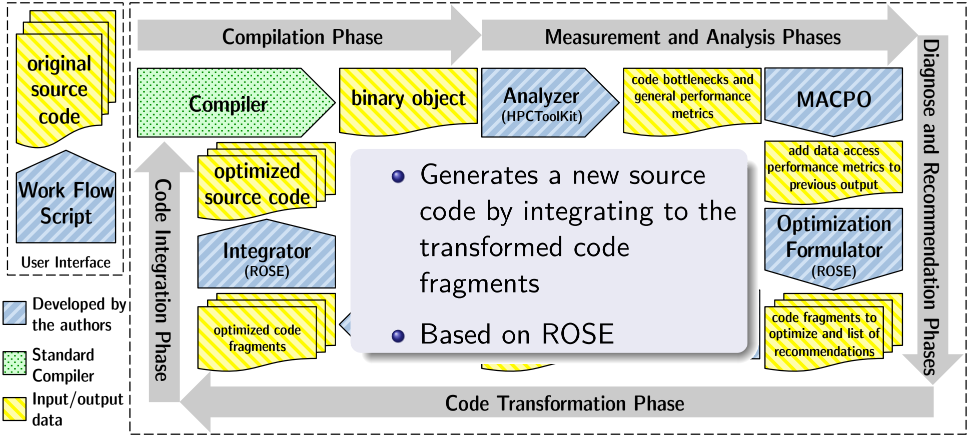

original source code
Work Flow Script
User Interface
Developed by the authors
Standard Compiler
Input/output data
Compilation Phase
Measurement and Analysis Phases
Diagnose and Recommendation Phases
Code Transformation Phase
Code Integration Phase
Compiler
binary object
Analyzer (HPCToolKit)
code bottlenecks and general performance metrics
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
code fragments to optimize and list of recommendations
Pattern Recognizer (Bison/Flex)
code fragments to optimize and list of code transformers
Transformer (PIPS/ROSE)
optimized code fragments
Integrator (ROSE)
optimized source code
Support Database

How PerfExpert does that: Integrator


original source code
Work Flow Script
User Interface
Developed by the authors
Standard Compiler
Input/output data
Compilation Phase
Measurement and Analysis Phases
Compiler
binary object
Analyzer (HPCToolKit)
code bottlenecks and general performance metrics
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
code fragments to optimize and list of recommendations
Diagnose and Recommendation Phases
Code Transformation Phase
Code Integration Phase
optimized source code
Integrator (ROSE)
optimized code fragments

- Generates a new source code by integrating to the transformed code fragments

How PerfExpert does that: Integrator


original source code
Work Flow Script
User Interface
Developed by the authors
Standard Compiler
Input/output data
Compilation Phase
Measurement and Analysis Phases
Compiler
binary object
Analyzer (HPCToolKit)
code bottlenecks and general performance metrics
MACPO
add data access performance metrics to previous output
Optimization Formulator (ROSE)
code fragments to optimize and list of recommendations
Diagnose and Recommendation Phases
Code Transformation Phase
optimized code fragments
Integrator (ROSE)
optimized source code
Code Integration Phase

- Generates a new source code by integrating to the transformed code fragments
- Based on ROSE

How PerfExpert does that: Key Points

How PerfExpert does that: Key Points

Why is this performance optimization “architecture” strong?

How PerfExpert does that: Key Points

Why is this performance optimization "architecture" strong?

- Each piece of the tool chain can be updated/upgraded individually

How PerfExpert does that: Key Points

Why is this performance optimization "architecture" strong?

- Each piece of the tool chain can be updated/upgraded individually
- It is flexible: you can add new metrics as well as plug new tools to measure application performance

How PerfExpert does that: Key Points

Why is this performance optimization "architecture" strong?

- Each piece of the tool chain can be updated/upgraded individually
- It is flexible: you can add new metrics as well as plug new tools to measure application performance
- **It is extendable**: new recommendations, transformations and strategies to select recommendations (we are counting on you!)

How PerfExpert does that: Key Points

Why is this performance optimization "architecture" strong?

- Each piece of the tool chain can be updated/upgraded individually
- It is flexible: you can add new metrics as well as plug new tools to measure application performance
- **It is extendable**: new recommendations, transformations and strategies to select recommendations (we are counting on you!)
- Multi-language, **multi-architecture**, open-source and built on top of well-established tools (HPCToolKit, ROSE, PIPS, etc.)

How PerfExpert does that: Key Points

Why is this performance optimization "architecture" strong?

- Each piece of the tool chain can be updated/upgraded individually
- It is flexible: you can add new metrics as well as plug new tools to measure application performance
- **It is extendable**: new recommendations, transformations and strategies to select recommendations (we are counting on you!)
- Multi-language, **multi-architecture**, open-source and built on top of well-established tools (HPCToolKit, ROSE, PIPS, etc.)
- Easy to use and lightweight!

Extending PerfExpert

Extending PerfExpert

Extending PerfExpert

- Adding performance metrics

Extending PerfExpert

- Adding performance metrics
- Optimization recommendations [entries on the SQL database]

Extending PerfExpert

- Adding performance metrics
- Optimization recommendations [entries on the SQL database]
- “Recommendation selection functions”

Extending PerfExpert

- Adding performance metrics
- Optimization recommendations [entries on the SQL database]
- “Recommendation selection functions”
- Pattern recognizers

Extending PerfExpert

- Adding performance metrics
- Optimization recommendations [entries on the SQL database]
- "Recommendation selection functions"
- Pattern recognizers
- Code transformers

Extending PerfExpert

Extending PerfExpert

Adding Performance Metrics

Extending PerfExpert

Adding Performance Metrics

```
code.section_info=Loop in function compute() at mm.c:8
code.filename=mm.c
code.line_number=8
code.type=loop
code.function_name=compute
code.extra_info=3
code.representativeness=99.8
perfexpert.ratio.data_accesses=0.25
perfexpert.instruction_accesses.L2i_hits=0.002
perfexpert.branch_instructions.mispredicted=0.0
perfexpert.floating-point_instr.fast_FP_instr=5.073
perfexpert.data_accesses.L2d_hits=1.846
...
```

Extending PerfExpert

Extending PerfExpert

Recommendation Selection Functions

Extending PerfExpert

Recommendation Selection Functions

```
SELECT r.id AS recommendation_id, SUM(
  (CASE c.short WHEN 'd-l1' THEN
    (m.data_accesses_L1d_hits - (max * 0.1))
    ELSE 0 END) +
    ...  )  AS score FROM recommendation AS r
JOIN metric AS m JOIN (SELECT MAX(
  m.data_accesses_L1d_hits, m.data_accesses_L2d_hits,
  ...  )  AS max
  FROM metric AS m WHERE m.overall * 100 /
    (0.5 * (100 - m.ratio_floating_point) +
    m.ratio_floating_point) > 1
  AND m.id = @RID)
WHERE (r.loop <= @LPD AND m.code_type = 'loop') OR
  (r.loop IS NULL AND m.code_type = 'function')
  AND m.id = @RID
GROUP BY r.id ORDER BY score DESC;
```

Extending PerfExpert

Extending PerfExpert

Pattern Recognizers

Extending PerfExpert

Pattern Recognizers

```
nested_iteration_statement
:  WHILE '(' exp ')' WHILE '(' exp ')' stmnt
| WHILE '(' exp ')' '' WHILE '(' exp ')' stmnt ''
| DO DO stmnt WHILE '(' exp ')' ';' stmnt WHILE '(' exp ')' ';'
| DO '' DO stmnt WHILE '(' exp ')' ';' '' WHILE '(' exp ')' ';'
| FOR '(' exp_stmnt exp_stmnt ')' FOR '(' exp_stmnt exp_stmnt ')' stmnt
| FOR '(' exp_stmnt exp_stmnt ')' '' FOR '(' exp_stmnt exp_stmnt ')' stmnt ''
| FOR '(' exp_stmnt exp_stmnt exp ')' FOR '(' exp_stmnt exp_stmnt exp ')' stmnt
| FOR '(' exp_stmnt exp_stmnt exp ')' '' FOR '(' exp_stmnt exp_stmnt exp ')' stmnt '' ;
```

Extending PerfExpert

Extending PerfExpert

Code Transformers

Extending PerfExpert

Code Transformers

```
create c_loop2 ../source/mm.c
activate INTERPROCEDURAL_SUMMARY_PRECONDITION
activate TRANSFORMERS_INTER_FULL
activate PRECONDITIONS_INTER_FULL
setproperty SEMANTICS_FIX_POINT_OPERATOR ‘‘derivative’’
module compute
apply LOOP_INTERCHANGE
loop_8
apply UNSPLIT[%PROGRAM]
close
quit
```

Hands on Tutorial

Accessing Stampede:

- `ssh login@stampede.tacc.utexas.edu`
- use the password that has been provided to you

Request a Compute Node:

- `./reserve`
- now we are ready to go...

Hands on Tutorial

Accessing Stampede:

- `cd 1`
- `perfexpert`
- `perfexpert -s mm.c mm`
- `grep -R "running time" *`
- `more mm.c`
- `more perfexpert-temp-zUKfkx7/1/fragments/new/mm.c`
- `perfexpert mm`
- `perfexpert -r 5 mm`
- `cd ../2`
- `perfexpert -m -s backprop.c backprop`

Agenda

Agenda

What we saw in the morning:

- Introduction and motivation
- What PerfExpert can provide to you?
- Demo
- How PerfExpert does that? (opening Pandora's box)
- Extending PerfExpert
- Hands on tutorial
- Morning closure

Agenda

What we saw in the morning:

- Introduction and motivation
- What PerfExpert can provide to you?
- Demo
- How PerfExpert does that? (opening Pandora's box)
- Extending PerfExpert
- Hands on tutorial
- Morning closure

What we will see in the afternoon:

How to enhance the application performance using memory access metrics (MAPCO)

Agenda

Short Demo

Short demo

Agenda

Performance Optimization by Mapping to Accelerators

Performance Optimization by Mapping to Accelerators

Performance Optimization by Mapping to Accelerators

Mapping of code segments to accelerators is becoming one of the most methods for optimizing the performance of an application

Performance Optimization by Mapping to Accelerators

Mapping of code segments to accelerators is becoming one of the most methods for optimizing the performance of an application

Problem: how to select those parts of an application which will benefit from execution on an accelerator?

Code Segments for SIMT/SIMD Execution

Code Segments for SIMT/SIMD Execution

Code Segments for SIMT/SIMD Execution

- Optimize for multicore chip execution — **PerfExpert, why?**

Code Segments for SIMT/SIMD Execution

- Optimize for multicore chip execution — **PerfExpert, why?**
- Identify time consuming kernels in code — **PerfExpert**

Code Segments for SIMT/SIMD Execution

- Optimize for multicore chip execution — **PerfExpert, why?**
- Identify time consuming kernels in code — **PerfExpert**
- Eliminate kernels not easily mappable for SIMT/SIMD execution — **How?**

Code Segments for SIMT/SIMD Execution

- Optimize for multicore chip execution — **PerfExpert, why?**
- Identify time consuming kernels in code — **PerfExpert**
- Eliminate kernels not easily mappable for SIMT/SIMD execution — **How?**
- Characterize the kernels suitable for SIMT/SIMD execution — **What properties?**

Code Segments for SIMT/SIMD Execution

- Optimize for multicore chip execution — **PerfExpert, why?**
- Identify time consuming kernels in code — **PerfExpert**
- Eliminate kernels not easily mappable for SIMT/SIMD execution — **How?**
- Characterize the kernels suitable for SIMT/SIMD execution — **What properties?**
- Rank appropriate kernels using the characteristics identified in the last step

Code Segments for SIMT/SIMD Execution

- Optimize for multicore chip execution — **PerfExpert, why?**
- Identify time consuming kernels in code — **PerfExpert**
- Eliminate kernels not easily mappable for SIMT/SIMD execution — **How?**
- Characterize the kernels suitable for SIMT/SIMD execution — **What properties?**
- Rank appropriate kernels using the characteristics identified in the last step
- Estimate cost of data movement

Code Segments for SIMT/SIMD Execution

- Optimize for multicore chip execution — **PerfExpert, why?**
- Identify time consuming kernels in code — **PerfExpert**
- Eliminate kernels not easily mappable for SIMT/SIMD execution — **How?**
- Characterize the kernels suitable for SIMT/SIMD execution — **What properties?**
- Rank appropriate kernels using the characteristics identified in the last step
- Estimate cost of data movement
- Look for refactorings that will enable leaving data on accelerator

Code Segments for SIMT/SIMD Execution

- Optimize for multicore chip execution — **PerfExpert, why?**
- Identify time consuming kernels in code — **PerfExpert**
- Eliminate kernels not easily mappable for SIMT/SIMD execution — **How?**
- Characterize the kernels suitable for SIMT/SIMD execution — **What properties?**
- Rank appropriate kernels using the characteristics identified in the last step
- Estimate cost of data movement
- Look for refactorings that will enable leaving data on accelerator
- Generate compiler annotations for translation of C/C++/Fortran to CUDA/OpenCL

Code Segments for SIMT/SIMD Execution

- Optimize for multicore chip execution — **PerfExpert, why?**
- Identify time consuming kernels in code — **PerfExpert**
- Eliminate kernels not easily mappable for SIMT/SIMD execution — **How?**
- Characterize the kernels suitable for SIMT/SIMD execution — **What properties?**
- Rank appropriate kernels using the characteristics identified in the last step
- Estimate cost of data movement
- Look for refactorings that will enable leaving data on accelerator
- Generate compiler annotations for translation of C/C++/Fortran to CUDA/OpenCL
- Suggest kernels needing new algorithms

Code Segments for SIMT/SIMD Execution

Code Segments for SIMT/SIMD Execution

Unsuitable Kernels

Code Segments for SIMT/SIMD Execution

Unsuitable Kernels

- Frequent TLB misses

Code Segments for SIMT/SIMD Execution

Unsuitable Kernels

- Frequent TLB misses
- High fraction of branches

Code Segments for SIMT/SIMD Execution

Unsuitable Kernels

- Frequent TLB misses
- High fraction of branches
- Cache conflicts across cores

Code Segments for SIMT/SIMD Execution

Unsuitable Kernels

- Frequent TLB misses
- High fraction of branches
- Cache conflicts across cores
- Irregular access strides for kernel data structures

Characterizing “Good” Kernels

Code Segments for SIMT/SIMD Execution

Unsuitable Kernels

- Frequent TLB misses
- High fraction of branches
- Cache conflicts across cores
- Irregular access strides for kernel data structures

Characterizing "Good" Kernels

- Computational intensity

Code Segments for SIMT/SIMD Execution

Unsuitable Kernels

- Frequent TLB misses
- High fraction of branches
- Cache conflicts across cores
- Irregular access strides for kernel data structures

Characterizing "Good" Kernels

- Computational intensity
- Pure "local" SPMD parallelism

Code Segments for SIMT/SIMD Execution

Unsuitable Kernels

- Frequent TLB misses
- High fraction of branches
- Cache conflicts across cores
- Irregular access strides for kernel data structures

Characterizing “Good” Kernels

- Computational intensity
- Pure “local” SPMD parallelism
- Streaming parallelism or vectorization

Code Segments for SIMT/SIMD Execution

Unsuitable Kernels

- Frequent TLB misses
- High fraction of branches
- Cache conflicts across cores
- Irregular access strides for kernel data structures

Characterizing "Good" Kernels

- Computational intensity
- Pure "local" SPMD parallelism
- Streaming parallelism or vectorization
- Regular access strides for data structures

Code Segments for SIMT/SIMD Execution

Unsuitable Kernels

- Frequent TLB misses
- High fraction of branches
- Cache conflicts across cores
- Irregular access strides for kernel data structures

Characterizing "Good" Kernels

- Computational intensity
- Pure "local" SPMD parallelism
- Streaming parallelism or vectorization
- Regular access strides for data structures
- Data reuse factor and data transfer volume

Code Segments for SIMT/SIMD Execution

Unsuitable Kernels

- Frequent TLB misses
- High fraction of branches
- Cache conflicts across cores
- Irregular access strides for kernel data structures

Characterizing "Good" Kernels

- Computational intensity
- Pure "local" SPMD parallelism
- Streaming parallelism or vectorization
- Regular access strides for data structures
- Data reuse factor and data transfer volume
- "Limited" recursion

Code Segments for SIMT/SIMD Execution

Code Segments for SIMT/SIMD Execution

Ranking “Good” Kernels

Code Segments for SIMT/SIMD Execution

Ranking "Good" Kernels

- Curve fit characteristics to speed-up measurements of kernels that have already been mapped

Code Segments for SIMT/SIMD Execution

Ranking “Good” Kernels

- Curve fit characteristics to speed-up measurements of kernels that have already been mapped
- Sort by values of characteristics in some chosen order

Code Segments for SIMT/SIMD Execution

Ranking “Good” Kernels

- Curve fit characteristics to speed-up measurements of kernels that have already been mapped
- Sort by values of characteristics in some chosen order
- Hold up your thumb?

Example

Example

Agenda

Thank You