Java™ Platform Performance on Multicore: Better Performance or Bigger Headache?

Azeem Shahjahan Jiva

Advanced Micro Devices
Java Labs

www.amd.com

TS-9363

Goal

Improved multi-threaded Java™ Platform Performance

Demonstrate how hardware can assist performance analysis.

Agenda

Analysis

Improvements

Tools

Pitfalls

Demo

Summary

Application Analysis

What are we measuring?

- What does the application do?
  - Call graph
  - Loaded classes
- How does the application perform?
  - I/O performance
  - Memory usage
  - Lock contention

Agenda

Analysis

Improvements

Tools

Pitfalls

Demo

Summary

Application Improvements

Why speed up the application?

- Customer complaint
    - Usually number one
- Not meeting requirements
    - Certain time constraints by design
- Response time is longer than expected
    - Minutes instead of seconds

Application Improvements

How are we going to do it?

- Throw more hardware at it
  - If application doesn’t scale this won’t work
- Tune Java™ Virtual Machine (JVM™) flags?
  - Poor algorithms are unaffected by flags
  - Band-aid solution not a real fix
- Better algorithm
  - 80/20 rule
  - Can the hardware help?

The terms “Java Virtual Machine” and “JVM” mean a Virtual Machine for the Java™ platform

Agenda

Analysis

Improvements

Tools

Pitfalls

Demo

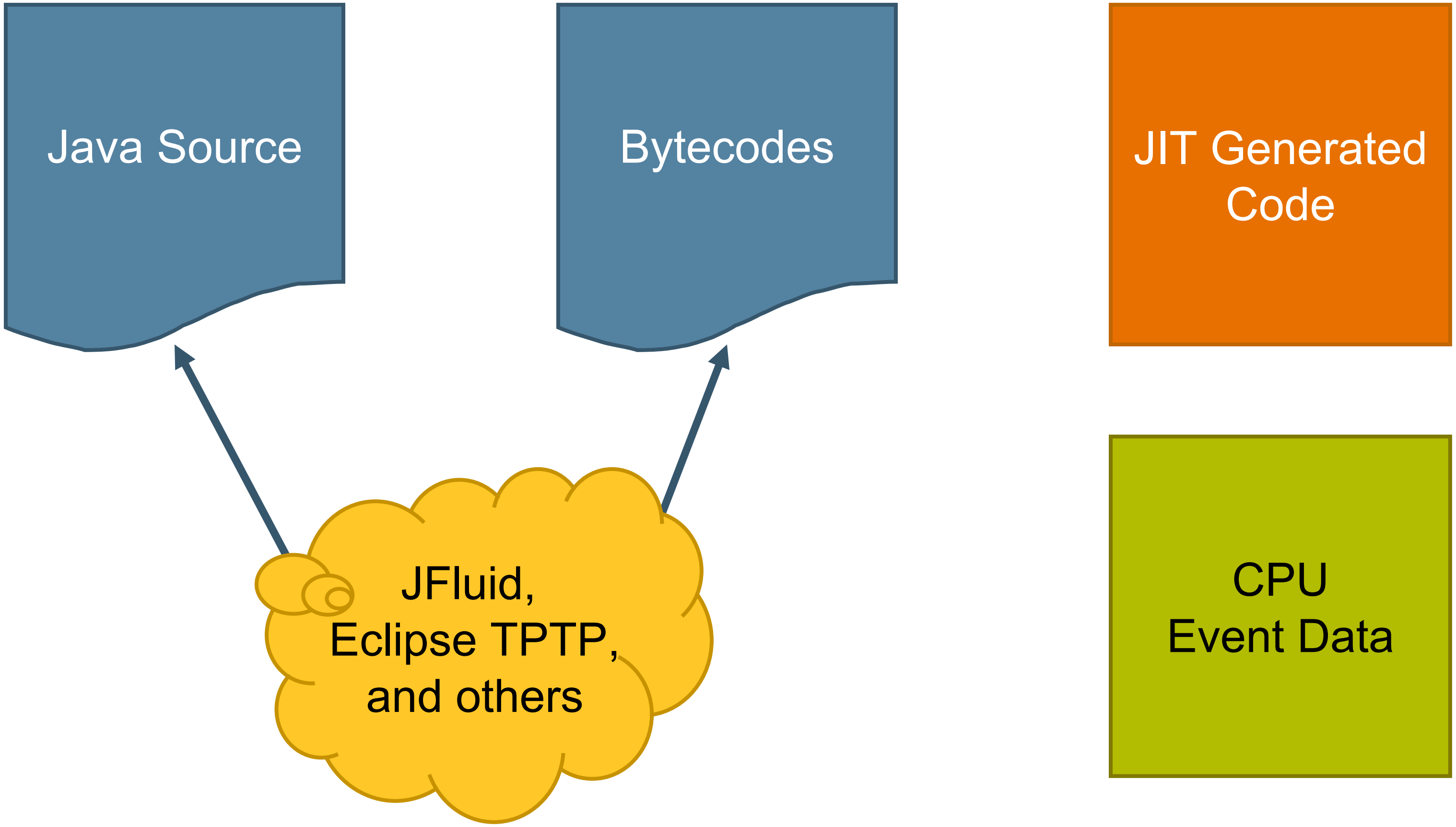
Summary

Many Tools Are Available

- Traditional Java programming language profilers
  - Bytecode instrumentation
    - Overhead is dependent on data collected
    - Flexible
  - VM generated events
    - Method entry/exit, Object allocated, etc.
    - Can break JVM virtual machine optimizations
  - Timer based
    - No JVM virtual machine information
    - Time spent in method
    - Low overhead

Tools

Traditional Java programming language profilers


Java Source

Bytecodes

JIT Generated Code

CPU Event Data

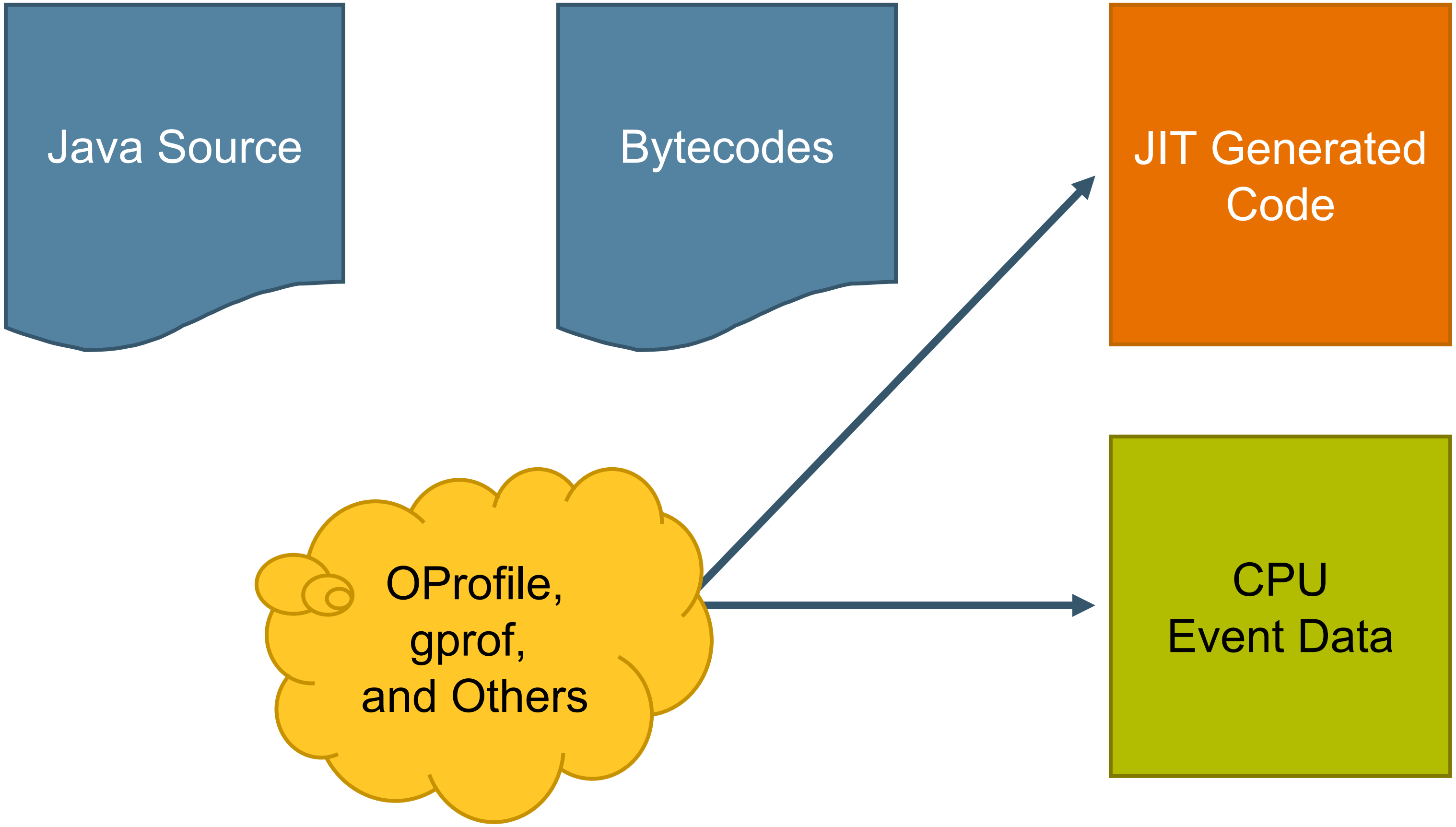
JFluid, Eclipse TPTP, and others

Many Tools Are Available

- What if we want to profile at hardware level?
  - Can the CPU help our profiling efforts?
- Tools available that can get CPU info
  - Need to map the data back to source code

Tools

Traditional hardware profilers



Java Source

Bytecodes

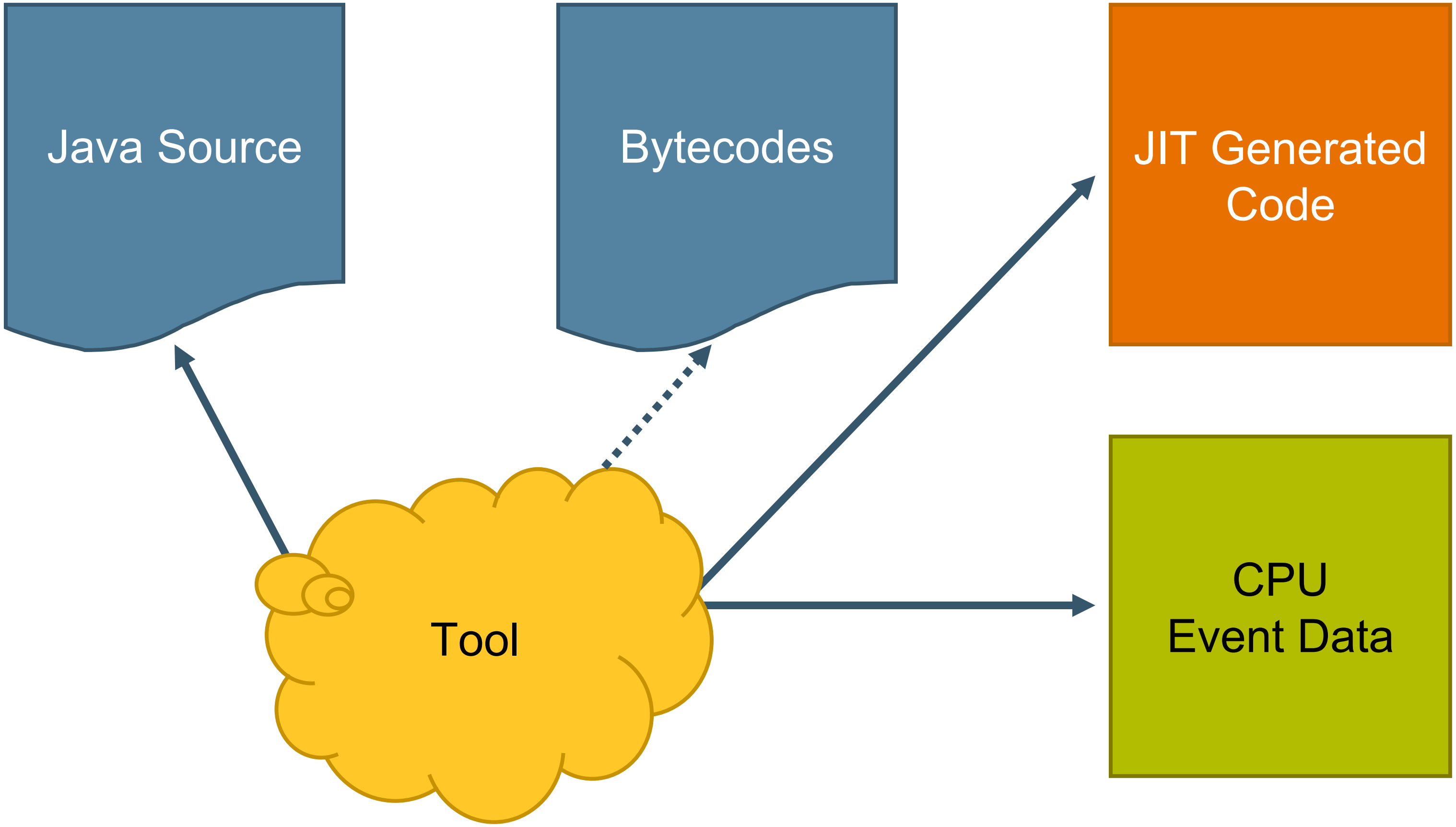
JIT Generated Code

CPU Event Data

OProfile, gprof, and Others

Tools

Combination of the two



AMD CodeAnalyst

What is it?

- Profiling tool
  - Timer-based profiling (TBP)
  - Event-based profiling (EBP)
  - Thread Analysis
  - Trace Generation and Pipeline Simulation
- Static code
  - C/C++
- Managed code
  - Java code
  - .NET

AMD CodeAnalyst

Why should I use it?

- Low overhead, system-wide profiling
- Works with applications that don’t work well with traditional profilers
- Supports multiple processor cores
- 32-bit and 64-bit Windows® and Linux®
- Free

Hardware Event Counters

What are they?

- Internal events within the processor
- Limited number of events at one time
- System wide
- Hardware information unavailable elsewhere
  - Set a trigger value
  - Interrupt when value is reached

Hardware Event Counters

What can I do with them?

- Interesting events
  - Cache hit/miss
  - Pipeline stalls
  - Decoder stalls
  - FPU stalls
  - Remote memory access

Agenda

Analysis

Improvements

Tools

Pitfalls

Demo

Summary

Profiling Pitfalls

Where can you go wrong?

- 80/20 rule
  - Improving code with most timer ticks first
- Realistic workload
  - Test programs don’t reflect reality
- Wrong events
  - Understand code before profiling
  - Pick events that match code
  - Trial and error

Profiling Pitfalls

Where can you go wrong?

- Profiling just once
  - Iterative process
  - Investigate one issue at a time
    - Cache access
    - Synchronization
- Investigating only user-written code
  - Issues can be in libraries
  - Rewrite code to use different library methods

DEMO

Java Platform Performance on Multicore: Better Performance or Bigger Headache?

Agenda

Analysis

Improvements

Tools

Pitfalls

Demo

Summary

Summary

- Threaded Java applications difficult to implement well
- Concurrency still an issue
  - TBP can help
  - Thread view can help
- Hardware events can help

Summary

- Constant improvements
  - 80/20 rule
- Use Hardware Events to investigate issues
- Algorithmic changes
- Improvements in code are processor agnostic

Q&A

Azeem Shahjahan Jiva

http://developer.amd.com/
http://www.amd.com/codeanalyst/

Trademark Attribution

AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Windows is a registered trademark of Microsoft Corporation. Linux is a registered trademark of Linus Torvalds. Other names used in this presentation are for identification purposes only and may be trademarks of their respective owners.

Java™ Platform Performance on Multicore: Better Performance or Bigger Headache?

Azeem Shahjahan Jiva

Advanced Micro Devices
Java Labs

www.amd.com

TS-9363