Analysis of Overhead in Dynamic Java Performance Monitoring

Vojtěch Horký, Jaroslav Kotrč, Peter Libič and Petr Tůma

Charles University in Prague
Context: Dynamic Monitoring of Production Systems
Dynamic Monitoring of Production Systems

Measurement probes are active only when needed, measuring everything all the time might not be practical.
Dynamic Monitoring of Production Systems

Measurement probes are active only when needed, measuring everything all the time might not be practical.
Measurement probes are active only when needed, measuring everything all the time might not be practical.

We are interested in performance of this function.
Dynamic Monitoring of Production Systems

Measurement probes are active only when needed, measuring everything all the time might not be practical.
Dynamic Monitoring of Production Systems

Measurement probes are active only when needed, measuring everything all the time might not be practical.

Code is dynamically instrumented when measuring.
Dynamic Monitoring of Production Systems

Measurement probes are active only when needed, measuring everything all the time might not be practical.

Once enough data is collected, probes are removed.
Issues of Dynamic Monitoring

In managed environments, code is compiled at run-time; probe insertion (removal) causes recompilation. Monitored application can thus behave differently.
Issues of Dynamic Monitoring

In managed environments, code is compiled at run-time; probe insertion (removal) causes recompilation. Monitored application can thus behave differently.

Interesting Questions

How do the code manipulations affect the application?
What is the overhead of such probe?
Is the observed performance representative?
Is there zero overhead once the probe is removed?
Experiment Setup
Experiment Coordination

- Application code
- Dynamic monitoring framework
- Measurements database
Experiment Coordination

- application code
- dynamic monitoring framework
- measurements database
- with static probes
- experiment coordination
- CPU & JVM monitoring
- measurements database
## Two Measurement Infrastructures

<table>
<thead>
<tr>
<th>Self-measurement</th>
<th>Dynamic monitoring</th>
</tr>
</thead>
</table>

- **Self-measurement**
  - Performance Baseline: "Observed"
  - Location: Method entry and exit points (both)

- **Dynamic monitoring**
  - Implementation: Native method (in C), Pure Java
  - Dynamic probe, static probe
  - Function of interest: static (self), measurement, dynamic measurement
# Two Measurement Infrastructures

<table>
<thead>
<tr>
<th></th>
<th>Self-measurement</th>
<th>Dynamic monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>Baseline</td>
<td>“Observed”</td>
</tr>
</tbody>
</table>

- **Method entry and exit points (both)**
- **Location**
- **Baseline**
- **Implementation**
  - Native method (in C)
  - Pure Java
- **Dynamic probe**
- **Static probe**
- **Function of interest**
- **Static (self) measurement**
- **Dynamic measurement**
## Two Measurement Infrastructures

<table>
<thead>
<tr>
<th>Performance</th>
<th>Self-measurement</th>
<th>Dynamic monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>Baseline</td>
<td>“Observed”</td>
</tr>
<tr>
<td></td>
<td>Method entry and exit points (both)</td>
<td></td>
</tr>
</tbody>
</table>
## Two Measurement Infrastructures

<table>
<thead>
<tr>
<th></th>
<th>Self-measurement</th>
<th>Dynamic monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Performance</strong></td>
<td>Baseline</td>
<td>“Observed”</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td>Method entry and exit points (both)</td>
<td></td>
</tr>
<tr>
<td><strong>Instrumentation</strong></td>
<td>Static</td>
<td>Dynamic (run-time)</td>
</tr>
</tbody>
</table>
# Two Measurement Infrastructures

<table>
<thead>
<tr>
<th></th>
<th>Self-measurement</th>
<th>Dynamic monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Performance</strong></td>
<td>Baseline</td>
<td>“Observed”</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td>Method entry and exit points (both)</td>
<td>Dynamic (run-time)</td>
</tr>
<tr>
<td><strong>Instrumentation</strong></td>
<td>Static</td>
<td>On demand</td>
</tr>
<tr>
<td><strong>Data collection</strong></td>
<td>Continuous</td>
<td></td>
</tr>
</tbody>
</table>
## Two Measurement Infrastructures

<table>
<thead>
<tr>
<th></th>
<th>Self-measurement</th>
<th>Dynamic monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Performance</strong></td>
<td>Baseline</td>
<td>“Observed”</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td>Method entry and exit points (both)</td>
<td></td>
</tr>
<tr>
<td><strong>Instrumentation</strong></td>
<td>Static</td>
<td>Dynamic (run-time)</td>
</tr>
<tr>
<td><strong>Data collection</strong></td>
<td>Continuous</td>
<td>On demand</td>
</tr>
<tr>
<td><strong>Implementation</strong></td>
<td>Native method (in C)</td>
<td>Pure Java</td>
</tr>
</tbody>
</table>
# Two Measurement Infrastructures

<table>
<thead>
<tr>
<th></th>
<th>Self-measurement</th>
<th>Dynamic monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Performance</strong></td>
<td>Baseline</td>
<td>“Observed”</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td>Method entry and exit points (both)</td>
<td>Dynamic (run-time)</td>
</tr>
<tr>
<td><strong>Instrumentation</strong></td>
<td>Static</td>
<td>Pure Java</td>
</tr>
<tr>
<td><strong>Data collection</strong></td>
<td>Continuous</td>
<td>On demand</td>
</tr>
<tr>
<td><strong>Implementation</strong></td>
<td>Native method (in C)</td>
<td></td>
</tr>
</tbody>
</table>

**Diagram:**

- Dynamic probe
- Static probe
- Function of interest
- Static measurement
- Dynamic measurement

**Images:**

- Department of Distributed and Dependable Systems (D3S) logo
Experiment Process
Experiment Process

run for some time
Experiment Process

- Run for some time
- Pick random method
- Run for some time
Experiment Process

- run for some time
- pick random method
- insert dynamic probe

Dynamic monitoring
Experiment Process

run for some time

pick random method

insert dynamic probe

Dynamic monitoring

run for some time
Experiment Process

- run for some time
- pick random method
- insert dynamic probe
- run for some time
- dump from static probes
- dump from dynamic probe
- Dynamic monitoring
Experiment Process

- run for some time
- pick random method
- insert dynamic probe
- run for some time
- dump from static probes
- dump from dynamic probe

What is the observed performance?
Experiment Process

run for some time

pick random method

insert dynamic probe

Dynamic monitoring

run for some time

dump from static probes

How fast it runs with dynamic monitoring?

What is the observed performance?

dump from dynamic probe
Experiment Process

- **run for some time**
- **pick random method**
- **run for some time**
- **insert dynamic probe**
- **dump from static probes**
- **remove dynamic probe**
- **dump from dynamic probe**
- **Dynamic monitoring**
  - How fast it runs **with dynamic monitoring**?
  - What is the **observed** performance?
Experiment Process

- **Flowchart Description**:
  - **Run for some time**
  - **Pick random method**
  - **Insert dynamic probe**
  - **Dynamic monitoring**
  - **Run for some time**
  - **Dump from static probes**
  - **Remove dynamic probe**
  - **Dump from dynamic probe**
  - **How fast it runs with dynamic monitoring?**
  - **What is the observed performance?**
Experiment Process

- run for some time
- pick random method
- insert dynamic probe
  - run for some time
  - dump from static probes
  - run for some time
  - remove dynamic probe
  - dump from dynamic probe

Dynamic monitoring

- How fast it runs with dynamic monitoring?
- run for some time
- dump from static probes
- dump from dynamic probe

What is the observed performance?
Experiment Process

1. Pick random method
2. Run for some time
3. Insert dynamic probe
4. Dump from static probes
5. Remove dynamic probe
6. Run for some time
7. Dump from dynamic probe
8. What is the observed performance?
9. Dump from static probes
10. How fast it runs with dynamic monitoring?
11. How fast it runs without dynamic monitoring?

Dynamic monitoring:

- Pick random method
- Run for some time
- Insert dynamic probe
- Dump from static probes
- Remove dynamic probe
- Run for some time
- Dump from dynamic probe
- What is the observed performance?
Experiment Process

- Run for some time
- Pick random method
- Insert dynamic probe
- Dump from static probes
- How fast it runs with dynamic monitoring?
- Dump from static probes
- Dump from dynamic probe
- Remove dynamic probe
- Run for some time
- What is the observed performance?

- Run for some time
- Dump from static probes
- How fast it runs without dynamic monitoring?
- Run for some time

- How fast it runs with dynamic monitoring?
- Dump from static probes
- Dump from dynamic probe
- What is the observed performance?
Platform and Application Details

Hardware: 32 CPUs, 2 NUMA nodes, 48G RAM.

SPECjbb2015 augmented with static probes.

Fixed request rate 4000 reqs/s. (Close to maximum with static probes on our hardware.)

Over 1200 monitored methods.

Business code of the benchmark.

Practically all methods called frequently enough.

About one minute of dynamic monitoring per method.

Several TBs of raw data per week of run-time.
Platform and Application Details

- Hardware: 32 CPUs, 2 NUMA nodes, 48G RAM.
Platform and Application Details

- **Hardware**: 32 CPUs, 2 NUMA nodes, 48G RAM.
- **SPECjbb2015** augmented with static probes.
  - Fixed request rate 4000 reqs/s.
  (Close to maximum with static probes on our hardware.)
- **Over 1200 monitored methods**.
  - Business code of the benchmark.
  - Practically all methods called frequently enough.
  - About one minute of dynamic monitoring per method.
Platform and Application Details

- Hardware: 32 CPUs, 2 NUMA nodes, 48G RAM.

- SPECjbb2015 augmented with static probes.
  - Fixed request rate 4 000 reqs/s.
    (Close to maximum with static probes on our hardware.)

- Over 1 200 monitored methods.
  - Business code of the benchmark.
  - Practically all methods called frequently enough.
  - About one minute of dynamic monitoring per method.

- Several TBs of raw data per week of run-time.
Results
Overall Overhead of Dynamic Monitoring
Overall Overhead of Dynamic Monitoring

Record CPU utilization \textbf{with} dynamic monitoring ...

... and \textbf{without} it.

dump from static probes

dump from dynamic probe

dump from static probes

remove dynamic probe

run for some time

run for some time

insert dynamic probe

pick random method
Measuring one method (even a hot one) at a time brings no significant overhead.
Measuring one method (even a hot one) at a time brings no significant overhead.
Time Needed for Just-in-time Recompilation
The JIT compiler typically needs at least 30 seconds to finish recompilations after probe insertion (removal).
JIT compiler typically needs at least 30 s to finish recompilations after probe insertion (removal).
Accuracy of Collected Data
Accuracy of Collected Data

- Run for some time
  - Pick random method
    - Insert dynamic probe
      - Dump from static probes
        - Run for some time
          - Dump from static probes
          - Dump from dynamic probe
          - Remove dynamic probe
  - Ratio between observed and baseline performance.
  - Dynamic monitoring
    - Run for some time
Accuracy of Collected Data

Ratio between times reported by static and dynamic probes

Method execution time (static probe) [s]
Accuracy of Collected Data

Ratio between times reported by static and dynamic probes

Method execution time (static probe) [µs]

Interpretation of numbers from dynamic monitoring:

- **Observed**
  - 50 µs
  - 20 µs
  - 2 µs

- **Actual**
  - 45 µs
  - 10 µs
  - 2 µs

Department of Distributed and Dependable Systems
Accuracy of Collected Data

Interpretation of numbers from dynamic monitoring:

<table>
<thead>
<tr>
<th>Observed</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 µs</td>
<td>45 µs – 50 µs</td>
</tr>
<tr>
<td>20 µs</td>
<td>10 µs – 20 µs</td>
</tr>
<tr>
<td>2 µs</td>
<td>$\frac{1}{2}$ µs – 2 µs</td>
</tr>
</tbody>
</table>

Method execution time (static probe) [µs]
Impact of Dynamic Monitoring
Impact of Dynamic Monitoring

- Pick random method
- Insert dynamic probe
- Dump from static probes
- Run for some time
- Dump from dynamic probe
- Remove dynamic probe
- Dump from static probes
- Run for some time

Ratio of baseline performance with and without dynamic monitoring
Impact of Dynamic Monitoring

Static probes: ratio of mean execution times during and after dynamic instrumentation

Faster when being monitored  Slower when being monitored

⇐⇒
Impact of Dynamic Monitoring

Faster when being monitored

Slower when being monitored

Static probes: ratio of mean execution times during and after dynamic instrumentation
Dynamic monitoring can observe shorter times (as if the probes speeded-up the application).
Conclusion
Analysis of Overhead
in Dynamic Java Performance Monitoring

We evaluated how dynamic monitoring affects a running application and what is the accuracy of the obtained data.

Rules of thumb coming from our experiment...

– Measuring one method at a time does not change CPU utilization.
– At least 30s are needed for (JIT) recompilation.
– If the reported time is 30 µs, the actual duration is between 20 µs and 40 µs (durations of at least 100 µs are more “trustworthy”, though).

Thank You!
Analysis of Overhead in Dynamic Java Performance Monitoring

We evaluated how dynamic monitoring affects a running application and what is the accuracy of the obtained data.

**Rules of thumb coming from our experiment . . .**

– Measuring one method at a time does not change CPU utilization.
– At least 30 s are needed for (JIT) recompilation.
– If the reported time is 30 µs . . .

  . . . the actual duration is between 20 µs and 40 µs

(durations of at least 100 µs are more “trustworthy”, though).
Analysis of Overhead in Dynamic Java Performance Monitoring

We evaluated how dynamic monitoring affects a running application and what is the accuracy of the obtained data.

Rules of thumb coming from our experiment . . .

– Measuring one method at a time does not change CPU utilization.
– At least 30 s are needed for (JIT) recompilation.
– If the reported time is 30 $\mu$s . . .

... the actual duration is between 20 $\mu$s and 40 $\mu$s

(durations of at least 100 $\mu$s are more “trustworthy”, though).

http://d3s.mff.cuni.cz/resources/icpe2016
Analysis of Overhead in Dynamic Java Performance Monitoring

We evaluated how dynamic monitoring affects a running application and what is the accuracy of the obtained data.

Rules of thumb coming from our experiment . . .

– Measuring one method at a time does not change CPU utilization.
– At least 30 s are needed for (JIT) recompilation.
– If the reported time is 30 µs . . .
  . . . the actual duration is between 20 µs and 40 µs
  (durations of at least 100 µs are more “trustworthy”, though).

http://d3s.mff.cuni.cz/resources/icpe2016

Thank You!