

0.1 Introdução

Este documento apresenta um resumo dos benchmarks realizados com a ferramenta *Java Microbenchmark Harness (JMH)*, descrevendo os tipos de testes e as variáveis utilizadas.

0.2 Código de Execução do Benchmark

```
package org.sample;

import org.openjdk.jmh.runner.Runner;
import org.openjdk.jmh.runner.options.Options;
import org.openjdk.jmh.runner.options.OptionsBuilder;

public class Main {
    public static void main(String[] args) throws Exception {
        Options opt = new OptionsBuilder()
            .include("BlackholeBench")
            .forks(1)
            .build();

        /**
         * new Runner(opt).run();
        */
    }
}
```

0.3 Tipos de Benchmarks

0.3.1 BlackholeBench

Declara variáveis: `byte`, `boolean`, `char`, `short`, `int`, `long`, `float`, `double`, `Object`.

Possui dois tipos de benchmark:

- **Implicit:** mede o tempo de acessar e retornar a variável. A JVM pode otimizar e eliminar o retorno se ele não for utilizado.
- **Explicit:** garante que o valor seja utilizado de forma que a JVM não possa otimizar, medindo o custo real de acesso.

0.3.2 BlackholeConsecutiveBench

Mede o tempo médio da JVM para executar operações simples.

Variáveis: `boolean`, `byte`, `short`, `char`, `int`, `float`, `long`, `double`, `Object`.

Cada variável possui três métodos:

- 1: consome uma operação;
- 4: consome quatro operações consecutivas;
- 8: consome oito operações consecutivas.

Observação: garante que o valor seja usado de forma que a JVM não possa otimizar. Os métodos 4 e 8 medem o custo de múltiplas operações consecutivas.

0.3.3 BlackholeConsumeCPUBench

```
public class BlackholeConsumeCPUBench {  
  
    ...  
  
    @Param("0")  
    private int delay;  
  
    @Benchmark  
    public void consume() {  
        Blackhole.consumeCPU(delay);  
    }  
    ...  
}
```

Descrição:

- Executa um loop vazio com `delay` iterações;
- Gasta CPU para simular trabalho;
- Mede o tempo médio gasto por chamada de `consumeCPU`.

0.3.4 BlackholePipelineBench

Variáveis: `boolean`, `byte`, `short`, `char`, `int`, `float`, `long`, `double`, `Object`, `arrays`.

Mede o custo de percorrer arrays de elementos consecutivos.

0.3.5 BlackholePipelinePayloadBench

Variáveis: `boolean`, `byte`, `short`, `char`, `int`, `float`, `long`, `double`, `objects`, `arrays`.

Mede o custo de percorrer arrays de elementos consecutivos, realizar uma operação e consumi-los.

0.3.6 BlackholeValueBench

Variáveis: `Boolean`

Mede o custo de percorrer arrays de elementos booleanos e consumi-los.

0.3.7 BurstStabilityBench

Mede o tempo que a CPU leva para realizar um trabalho simulado.

0.3.8 CompilerHintsBench

Calcula o logaritmo de PI, com quatro métodos diferentes:

- do_Plain método normal.
- do_Inline instrução ao compilador: tente sempre fazer inline deste método.
- do_DontInline instrução: não faça inline deste método.
- do_Exclude instrução: ignore este método para otimizações JIT.

Inline significa que o compilador pode substituir a chamada ao método pelo corpo do método, evitando overhead de chamada.

Mostra a influência do inlining.

0.3.9 CoreStabilityBench

0.4 Resumo em Tabela

| Benchmark | Tipo de Variáveis | Descrição |
|--------------------------------|----------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| BlackholeBench | byte, boolean, char, short, int, long, float, double, Object | Implicit: acesso otimizado; Explicit: acesso real sem otimização |
| BlackholeConsecutiveBench | boolean, byte, short, char, int, float, long, double, Object | Mede custo de 1, 4 ou 8 operações consecutivas, garantindo uso de valores |
| BlackholeConsumeCPU-Bench | | Executa loop vazio para simular CPU Mede tempo médio de consumo CPU |
| BlackholePipelineBench | boolean, byte, short, char, int, float, long, double, Object, Array | Mede custo de percorrer arrays de elementos consecutivos |
| BlackholePipelinePayload-Bench | boolean, byte, short, char, int, float, long, double, Object, arrays | Mede custo de percorrer arrays de elementos consecutivos, realizar operações e consumi-los. |
| BlackholeValueBench | boolean | Mede o custo de percorrer arrays de elementos booleanos e consumi-los. |
| BurstStabilityBench | | Mede o tempo que a CPU leva para realizar um trabalho simulado. |
| CompilerHintsBench | | Mostra a influência do inlining. |