benchmark/example_01

writ3it

29 marca 2019

Performance comparison of object creation with “variable” count of arguments

PHP Version

Copyright (c) 1997-2018 The PHP Group Zend Engine v3.3.3, Copyright (c) 1998-2018 Zend Technologies with Zend OPcache v7.3.3-1+ubuntu18.04.1+deb.sury.org+1, Copyright (c) 1999-2018, by Zend Technologies

Methods

Static call

Method that found at legacy code by member of polish php community.

```
function call($argv, $args){
    $class = Foo::class;
    switch($argv){
        case 0:
            return new $class();
        case 1:
            return new $class($args[0]);
        case 2:
            return new $class($args[0],$args[1]);
        case 3:
            return new $class($args[0],$args[1],$args[2]);
        case 4:
            return new $class($args[0],$args[1],$args[2],$args[3]);
        case 5:
            return new $class($args[0],$args[1],$args[2],$args[3],$args[4]);
        case 6:
            return new $class($args[0],$args[1],$args[2],$args[3],$args[4],$args[5]);
        case 7:
            return new $class($args[0],$args[1],$args[2],$args[3],$args[4],$args[5],$args[6]);
        case 8:
            return new $class($args[0],$args[1],$args[2],$args[3],$args[4],$args[5],$args[6],$args[7]);
    };
    return ;
}
```

Reflection

Nobody like reflection because it’s very slow.

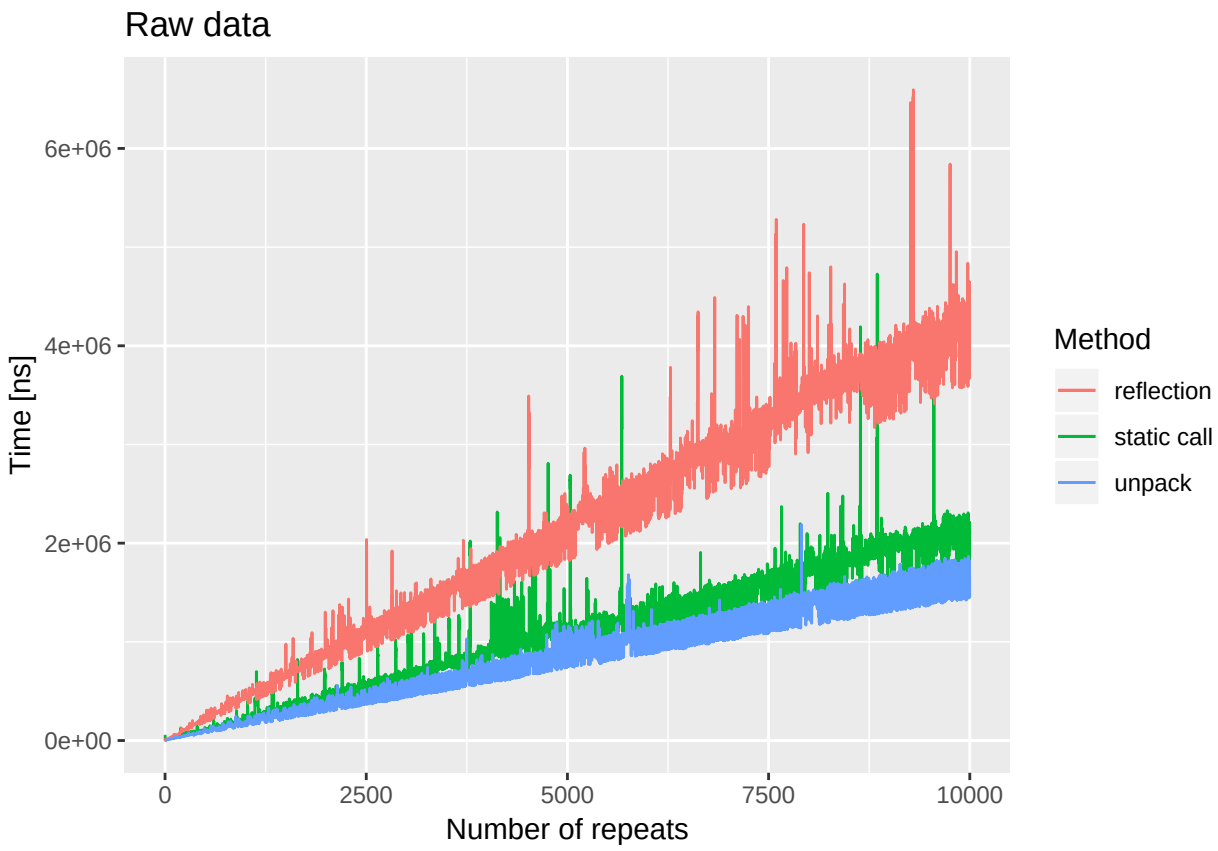
```
    $reflect  = new ReflectionClass(Foo::class);
    return $reflect->newInstanceArgs($args);
```

Unpack (splat operator)

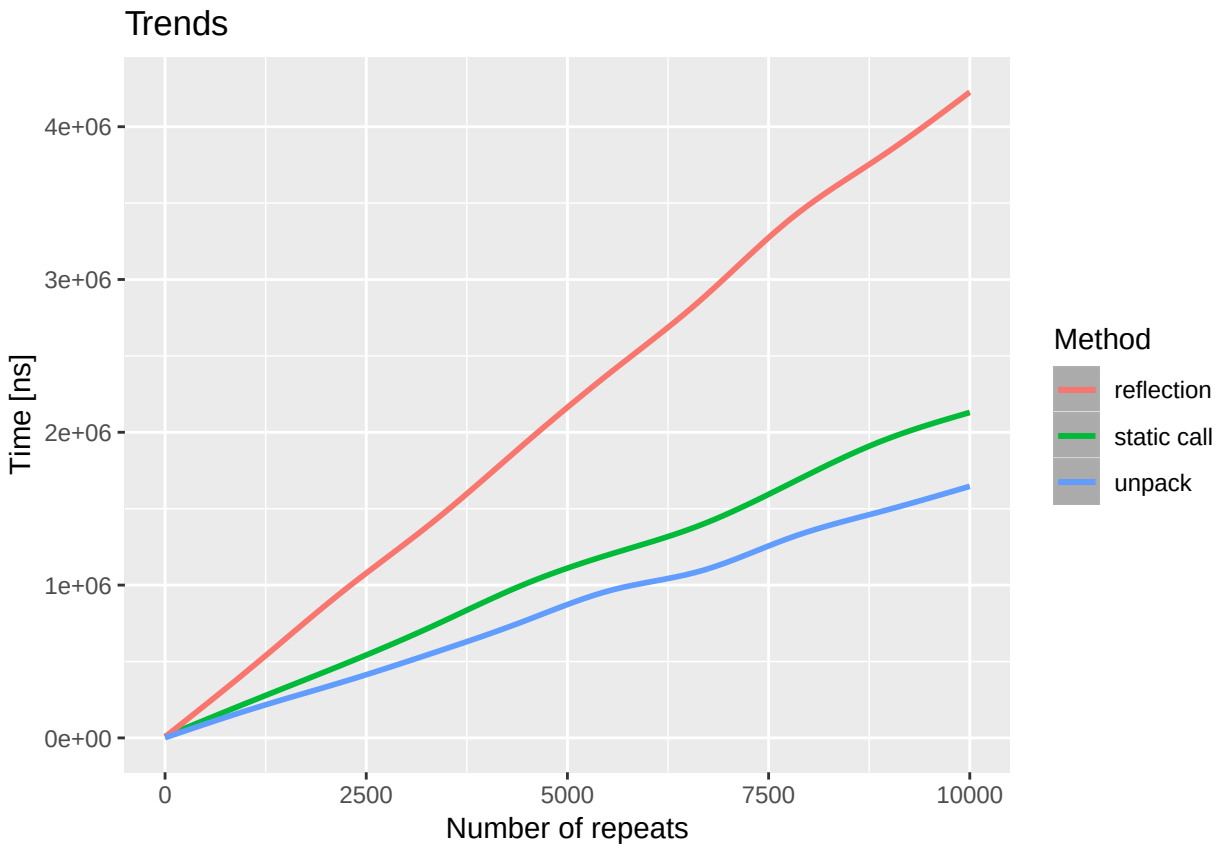
Splat operator tells interpreter to use as next arguments correspondings element in array.

```
    $method = Foo::class;
    return new $method(...$args);
```

Comparison charts




```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Conclusion

The most performance way is to use splat operator.