对比在不使用 redis 缓存和使用 redis 缓存两种情况下服务器性能。
【1】首先在本地建个数据库、表。

```
CREATE DATABASE db_redis;
USE db_redis;
CREATE TBALE test(
id int primary key auto_increment,
content varchar(20)
);
```

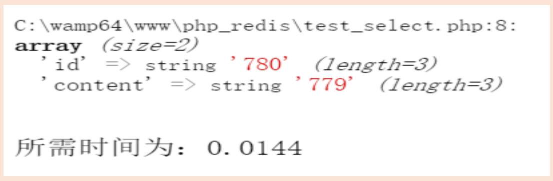
【2】向 test 数据表中添加 1000 行数据，填充数据的代码如下：

```
<?php
	$mysqli = new mysqli('localhost','root','','db_redis');
	if($mysqli->connect_errno){
		die('connect error:'.$mysqli->connect_error);
	}else{
		$mysqli->set_charset('utf8');
	}
	for($i=0;$i<1000;$i++){
		$sql = "INSERT INTO test(content) VALUES($i)";
		$mysqli->query($sql);
	}
	$mysqli->close();
```

【3】从数据表中随机获得一个数据，代码如下：

```
<?php
	$time1 = microtime();//当前 Unix 时间戳的微秒数
	require_once 'conn.php';
	$i = rand(1,1000);
	$sql = "SELECT *FROM test WHERE id=".$i;
	$res = $mysqli->query($sql);
	$out = $res->fetch_assoc();
	var_dump($out);
	$time2 = microtime();
	$time = $time2-$time1;
	echo "</br>所需时间为：";
	echo $time;
```

【4】执行上面的代码，获得一个随机数字显示的输出时间为：0.0144s 左右。



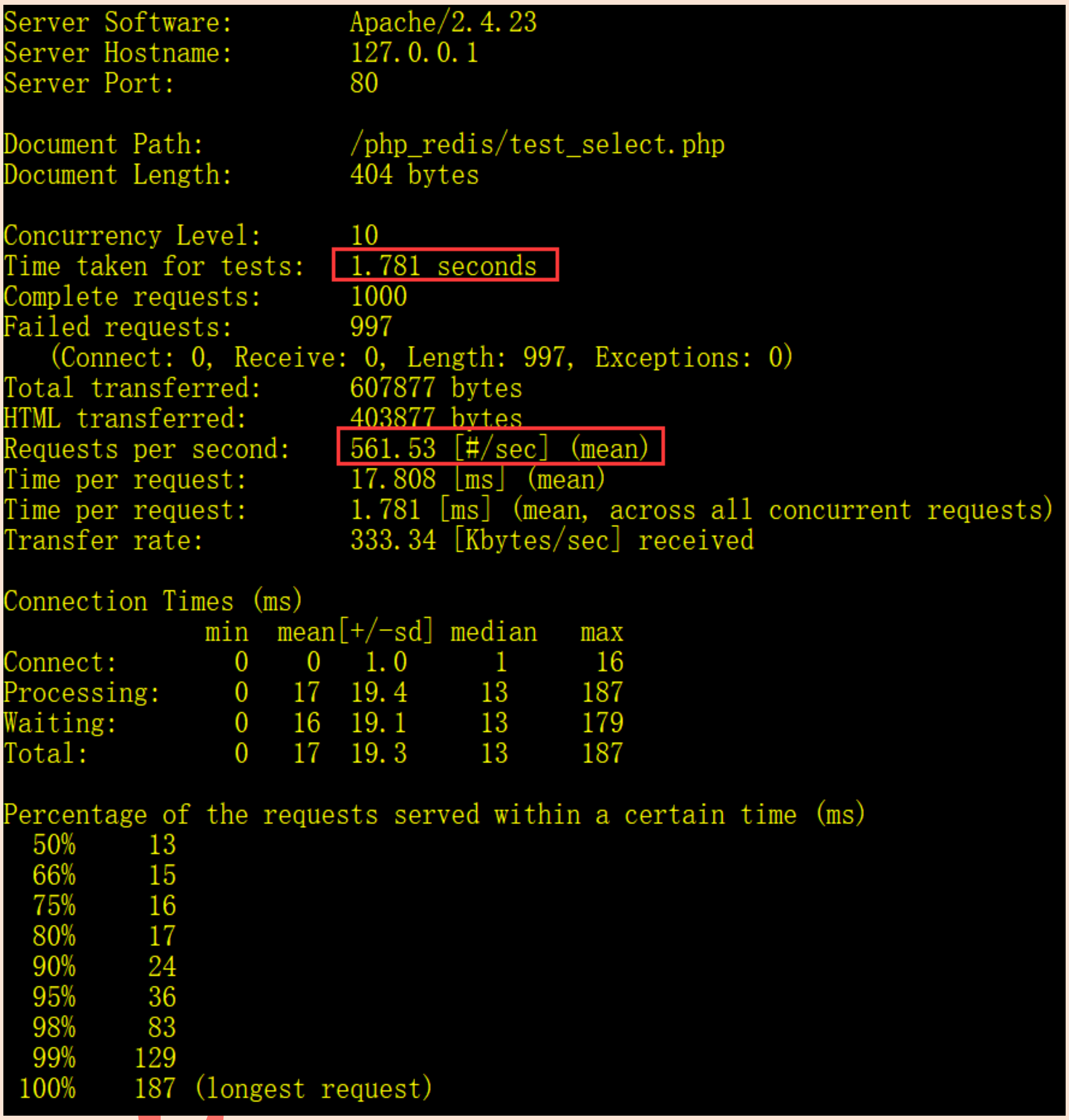
```
C:\wamp64\www\php_redis\test_select.php:8:
array (size=2)
  'id' => string '780' (length=3)
  'content' => string '779' (length=3)

所需时间为：0.0144
```

【5】然后使用 AB 工具（Apache 自带的网页压力测试工具）对其发起一个请求量为 1000，并发量为 10 的测试。

C:\Users\chans>cd C:\wamp64\bin\apache\apache2.4.23\bin

AB -n1000 -c10 http://127.0.0.1/php_redis/test_select.php



```
Server Software:        Apache/2.4.23
Server Hostname:        127.0.0.1
Server Port:            80

Document Path:          /php_redis/test_select.php
Document Length:        404 bytes

Concurrency Level:      10
Time taken for tests:   1.781 seconds
Complete requests:      1000
Failed requests:        997
   (Connect: 0, Receive: 0, Length: 997, Exceptions: 0)
Total transferred:      607877 bytes
HTML transferred:       403877 bytes
Requests per second:    561.53 [#/sec] (mean)
Time per request:       17.808 [ms] (mean)
Time per request:       1.781 [ms] (mean, across all concurrent requests)
Transfer rate:          333.34 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   1.0      1      16
Processing:     0   17  19.4     13     187
Waiting:        0   16  19.1     13     179
Total:          0   17  19.3     13     187

Percentage of the requests served within a certain time (ms)
  50%     13
  66%     15
  75%     16
  80%     17
  90%     24
  95%     36
  98%     83
  99%    129
 100%    187 (longest request)
```

测试结果可知，每秒处理请求数是 561.53，总耗时 1.781s。处理一个请求的平均时间为 17.808ms。

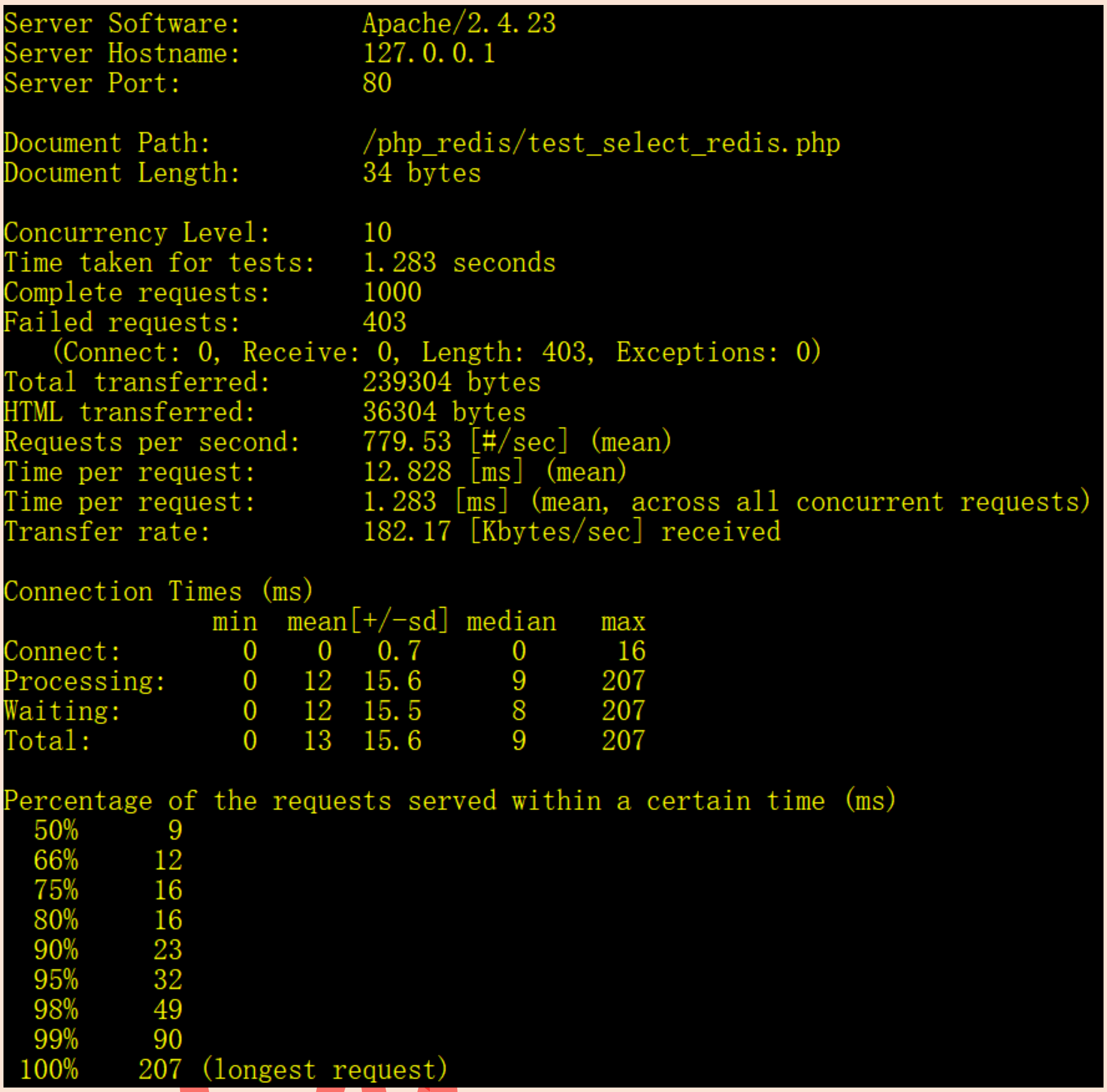
【6】接下来使用 redis 作为缓存，修改上述代码再次进行测试。代码如下：

```
<?php
	$time1 = microtime();//当前 Unix 时间戳的微秒数
	$redis = new Redis();
	$redis->connect("127.0.0.1","6379");
	$i=rand(1,1000);
	if($redis->get($i)){
		echo $redis->get($i);
	}else{
		require_once 'conn.php';
		$i = rand(1,1000);
		$sql = "SELECT *FROM test WHERE id=".$i;
		$res = $mysqli->query($sql);
		$out = $res->fetch_assoc();
		$arr = $out['content'];
		$redis->set($i,$arr);
		echo $redis->get($i);
	}
	$time2 = microtime();
	$time = $time2-$time1;
	echo "</br>所需时间为：".$time;
```

上面的代码执行时候，先检查 redis 中是否有所需数据，如果有则直接输出，如果没有则请求 MySQL 数据库获取数据，并将数据写入 redis，然后再从 redis 里读取数据。执行以上代码显示输出时间为 0.030509s,为什么使用 redis 缓存后时间反而慢了呢？这是因为我们随机取得的数据不在 redis 中，需要先向 MySQL 获取数据，这一步耗时较多。当再次执行代码时，如果获得的随机数已经成为 redis 里面的索引，则代码执行时间缩短，再次测试为：0.006772s 左右。

【7】同样，下面使用 AB 工具对该网页发起同样的测试。

```
C:\Users\chans>cd C:\wamp64\bin\apache\apache2.4.23\bin
               AB -n1000 -c10 http://127.0.0.1/php_redis/test_select_redis.php
```



```
Server Software:        Apache/2.4.23
Server Hostname:        127.0.0.1
Server Port:            80

Document Path:          /php_redis/test_select_redis.php
Document Length:        34 bytes

Concurrency Level:      10
Time taken for tests:   1.283 seconds
Complete requests:      1000
Failed requests:        403
   (Connect: 0, Receive: 0, Length: 403, Exceptions: 0)
Total transferred:      239304 bytes
HTML transferred:       36304 bytes
Requests per second:    779.53 [#/sec] (mean)
Time per request:       12.828 [ms] (mean)
Time per request:       1.283 [ms] (mean, across all concurrent requests)
Transfer rate:          182.17 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.7      0      16
Processing:     0   12  15.6      9     207
Waiting:        0   12  15.5      8     207
Total:          0   13  15.6      9     207

Percentage of the requests served within a certain time (ms)
  50%      9
  66%     12
  75%     16
  80%     16
  90%     23
  95%     32
  98%     49
  99%     90
 100%    207 (longest request)
```

测试结果可知，每秒处理请求数是 779.53，总耗时 1.283s。处理一个请求的平均时间为 12.828ms。

对于大数据量，效果更佳明显。