nodejs如何获取时间戳与时间差

Nodejs中获取时间戳的方法有很多种，例如：

1.**new Date().getTime()**

2.**Date.now()**

3.**process.uptime()**

4.**process.hrtime()**

平时想获取一个时间戳的话，用这些方法都可以,那么这些方法有什么区别呢？

new Date().getTime()和Date.now()

这些方法是通过node运行环境的系统时间毫秒数, **+new Date()** 写法的效果和 **new Date().getTime()** 效果相同。

在需要频繁使用时间戳的场景中，需要关注方法性能，这几种方法中 **Date.now()** 的性能最佳，可以通过一点代码来测试：

```
var t1 = new Date().getTime();
var t2 = t1;
var i = 0, count = 10000000, interval = 0;

for(i = 0; i < count; i++)
{
t2 = new Date().getTime();
interval = (t2 - t1);
}
interval = (t2 - t1);
console.log('【new Date().getTime()】interval: ', interval);

t1 = new Date().getTime();
for(i = 0; i < count; i++)
{
t2 = +new Date;
interval = (t2 - t1);
}
interval = (t2 - t1);
console.log('【+new Date】interval: ', interval);

t1 = new Date().getTime();
for(i = 0; i < count; i++)
{
t2 = Date.now();
interval = (t2 - t1);
}
interval = (t2 - t1);
console.log('【Date.now()】interval: ', interval);
```

输出结果：

【new Date().getTime()】interval: 1583

【+new Date】interval: 2189

【Date.now()】interval: 891

如果只是获取时间戳，那么使用**Date.now()**是最佳的做法，但是如果要计算时间差，这几个方法就会有点问题：运行环境的系统时间有时候是会有微小回调的，这样得到的时间差就不精确了，有时候会引发某些BUG。

process.hrtime()

这种方式是根据任意取的一个过去的时间点，距离现在的时间来获取一个精确的时间戳对象：[秒, 纳秒]

```
> process.hrtime()
[ 3197146, 563552237 ]
```

这种方式和系统时间无关，因此不会受到系统时钟漂移的影响，用来计算时间差的时候就不会有BUG了。

但是，万事总有但是 – –

如果用在一个被频繁调用的地方呢？

```
var t1 = new Date().getTime();
var t2 = t1;
var i = 0, count = 10000000, interval = 0;

var hrTime1 = process.hrtime();
var hrTime2 = hrTime1;

t1 = new Date().getTime();
for(i = 0; i < count; i++)
{
hrTime2 = process.hrtime(hrTime1);
}
t2 = new Date().getTime();
interval = parseInt(hrTime2[0] * 1e3 + hrTime2[1] * 1e-6);
console.log('【hrTime】interval: ', interval, t2 - t1);
```

【hrTime】interval: 6412 6413 没有记错的话，相同的创建次数，上面的**Date.now()**可是900ms左右啊！

process.hrtime()也太慢了有木有！！！

原来nodejs处理高精度时间的时候，计算比较复杂，占用系统资源多，速度慢，那么在高频应用的地方就不适合用这个方法了。下面请看**process.uptime()**

process.uptime（ ）

此函数是通过nodejs启动运行时间来得到一个秒数时间戳，精确到毫秒：

process.uptime

输入：6.419

此函数以node启动时间为准，同样也不会受系统时钟漂移影响，适合用来计算时间差。

那么多次调用性能如何呢？

```
var t1 = new Date().getTime();
var t2 = t1;
var i = 0, count = 10000000, interval = 0;

t1 = process.uptime()*1000;
for(i = 0; i < count; i++)
{
t2 = process.uptime()*1000;
//interval = (t2 - t1);
}
interval = (t2 - t1);
console.log('【process.uptime()】interval: ', interval);
```

输出：【process.uptime()】interval: 954

和**process.hrtime()**相比性能就搞出很多了~

不用算那么精确，就是快！

那么需要高频计算时间差的场合，就是你了！

以上就是nodejs获取时间戳与时间差的全部内容，希望对大家平时使用nodejs的时候能有所帮助。