

[首页](#) | [技术频道](#) | [51CTO旗下网站](#) | [地图](#)
[登录](#) | [注册](#) | [招聘](#) | [学院](#) | [下载](#) | [论坛](#) | [博客](#) | [更多](#)

网络安全一个大写的反“作死”
Android开发120天完成9个APP项目
微软企业级内训课程免费学
2016年备战软考-重难点解析

移动开发

[首页](#) | [Android](#) | [iOS](#) | [Windows Phone](#) | [BlackBerry](#) | [webOS](#) | [Symbian](#) | [bada](#) | [OPhone](#) | [其他](#)

请输入关键字

搜索

 您所在的位置: [移动开发](#) > [热门推荐](#) > [GCD实战二:资源竞争](#)

GCD实战二:资源竞争

2013-07-15 16:55 佚名 dreamingwish 字号: T | T

收藏 +

GCD是个比较范特西的技术，可以办到很多事儿，但是它不能为你办所有的事儿。所以，对于进行IO操作并且可能会使用大量内存的任务，我们必须仔细斟酌。当然，即使这样，GCD还是为我们提供了简单有效的方法来进行并发计算。

 AD: [51CTO 网+ 第十二期沙龙: 大话数据之美_如何用数据驱动用户体验](#)


概述

我将分四步来带大家研究研究程序的并发计算。第一步是基本的串行程序，然后使用GCD把它并行计算化。如果你想顺着步骤来尝试这些程序的话，可以下载源码。注意，别运行 imagegcd2.m，这是个反面教材。

 源码下载: <http://down.51cto.com/data/872222>

原始程序

我们的程序只是简单地遍历~/Pictures然后生成缩略图。这个程序是个命令行程序，没有图形界面（尽管是使用Cocoa开发库的），主函数如下：

```
1.  int main(int argc, char **argv)
2.  {
3.      NSAutoreleasePool *outerPool = [NSAutoreleasePool new];
4.      NSApplicationLoad();
5.      NSString *destination = @"/tmp/imagegcd";
6.      [[NSFileManager defaultManager] removeItemAtPath: destination error: NU
7.      LL];
8.      [[NSFileManager defaultManager] createDirectoryAtPath: destination
9.      IntermediateDirectories: YES
10.      attributes: nil
11.      error: NULL];
12.      Start();
13.      NSString *dir = [@"~/Pictures" stringByExpandingTildeInPath];
14.      NSDirectoryEnumerator *enumerator = [[NSFileManager defaultManager] enu
```


 关注有礼
51CTO官方微信
weixin51cto

 3万免费IT视频教程
随时随地学习


专题 Android Studio上手攻略



既然强大的Android Studio来了，有什么理由不去用呢？

[iOS新语言swift上手指南](#)
[MIUI 6 测评: 细节的量变](#)

文章排行

24小时

本周

本月

21个免费的UI界面设计工具、资源及网站
在Eclipse下搭建Android开发环境教程
人人都是开发者: 5款傻瓜式APP开发工具
三大移动Web开发框架哪个适合你?
哪个市场最挣钱, 腾讯告诉你!
阿里: 这是全世界最大的打假团队
Android应当从iOS处窃取的五项最佳功能
iOS 8出色的跨应用通信效果: 解读Actio
看库克如何干掉Android
为什么说2015年将是微服务架构元年?

热点职位

更多>>

移动客户端研发工程师

全职/1-3年/本科 12k-25k

多盟

SDK开发工程师

全职/1-3年/大专 7k-10k

游众游戏

```

meratorAtPath: dir];
14.     int count = 0;
15.     for (NSString *path in enumerator)
16.     {
17.         NSAutoreleasePool *innerPool = [NSAutoreleasePool new];
18.         if ([[path pathExtension] lowercaseString] isEqual: @"jpg"])
19.         {
20.             path = [dir stringByAppendingPathComponent: path];
21.
22.             NSData *data = [NSData dataWithContentsOfFile: path];
23.             if (data)
24.             {
25.                 NSData *thumbnailData = ThumbnailDataForData(data);
26.                 if (thumbnailData)
27.                 {
28.                     NSString *thumbnailName = [NSString stringWithFormat: @
"%d.jpg", count++];
29.                     NSString *thumbnailPath = [destination stringByAppending
PathComponent: thumbnailName];
30.                     [thumbnailData writeToFile: thumbnailPath atomically: N
O];
31.                 }
32.             }
33.         }
34.         [innerPool release];
35.     }
36.     End();
37.     [outerPool release];
38. }

```

如果你要看到所有的副主函数的话，到文章顶部下载源代码吧。当前这个程序是imagegcdl.m。程序中重要的部分都在这里了。Start 函数和 End 函数只是简单的计时函数（内部实现是使用的gettimeofday函数）。ThumbnailDataForData函数使用Cocoa库来加载图片数据生成Image对象，然后将图片缩小到320×320大小，最后将其编码为JPEG格式。

简单而天真的并发

乍一看，我们感觉将这个程序并发计算化，很容易。循环中的每个迭代器都可以放入GCD global queue中。我们可以使用dispatch queue来等待它们完成。为了保证每次迭代都会得到唯一的文件名数字，我们使用OSAtomicIncrement32来原子操作级别的增加count数：

```

1.     dispatch_queue_t globalQueue = dispatch_get_global_queue(0, 0);
2.     dispatch_group_t group = dispatch_group_create();
3.     __block uint32_t count = -1;
4.     for (NSString *path in enumerator)
5.     {
6.         dispatch_group_async(group, globalQueue, BlockWithAutoreleasePool(^
{
7.             if ([[path pathExtension] lowercaseString] isEqual: @"jpg"))
8.             {
9.                 NSString *fullPath = [dir stringByAppendingPathComponent: p
ath];
10.
11.                 NSData *data = [NSData dataWithContentsOfFile: fullPath];
12.                 if (data)
13.                 {
14.                     NSData *thumbnailData = ThumbnailDataForData(data);
15.                     if (thumbnailData)
16.                     {
17.                         NSString *thumbnailName = [NSString stringWithForma
t: @"%d.jpg",
18. OSAtomicIncrement32(&count)];
19.                         NSString *thumbnailPath = [destination stringByAppen
dingPathComponent: thumbnailName];
20.                         [thumbnailData writeToFile: thumbnailPath atomicall
y: NO];
21.                     }

```

APP开发工程师

全职/1-3年/本科 10k-20k 浙江长龙航空

Android软件工程师

全职/3-5年/大专 6k-10k 广州太乙科技

安卓工程师

全职/3-5年/大专 4k-8k 索引教育

热点专题

[更多>>](#)


iOS开发之常见疑难问题

在iOS开发过程中，尤其是对于新手来说，都会遇到或多



Web App开发最佳实践

Web App开发中会面临越来越“重”的问题，如果在开始



Android开发常见“疑

作为Android开发者，最头疼是什么？相信大家会异口同

热点标签

iOS开发 Android开发 Symbian开发 MeeGo开发
BlackBerry开发 Windows Phone开发 Phone Club
Android 4.0 webOS 3.0 智能手机 软件下载

[点击这里查看样刊](#)
[立即订阅](#)

全站热点



2013上半年移动互联网大会回顾



“棱镜”惊醒中国信息安全最弱神经

我国成功研制世界首台拟态计算机 能
2013扁平化设计终极指南
BYOD崛起之路 MBaaS满足企业移动应用
开发频道2013年9月第3周重点内容推荐
内存难题或挑战英特尔百亿亿次目标

读书



计算机网络安全

本书从计算机网络安全的概念入手，分析了单机节点、单一网络、互联网络和开放互联网络的基本安全问题，并对计算机网络安全体系架

```
22.         }
23.     }
24.     });
25. }
26.     dispatch_group_wait(group, DISPATCH_TIME_FOREVER);
```

这个就是imagegcd2.m，但是，注意，别运行这个程序，有很大的问题。

如果你无视我的警告还是运行这个imagegcd2.m了，你现在很有可能是在重启了电脑后，又打开了我的页面。。如果你乖乖地没有运行这个程序的话，运行这个程序发生的情况就是（如果你有很多很多图片在~/Pictures中）：电脑没反应，好久好久都不动，假死了。。

问题在哪

问题出在哪？就在于GCD的智能上。GCD将任务放到全局线程池中运行，这个线程池的大小根据系统负载来随时改变。例如，我的电脑有四核，所以如果我使用GCD加载任务，GCD会为我每个cpu核创建一个线程，也就是四个线程。如果电脑上其他任务需要进行的话，GCD会减少线程数来使其他任务得以占用cpu资源来完成。

但是，GCD也可以增加活动线程数。它会在其他某个线程阻塞时增加活动线程数。假设现在在四个线程正在运行，突然某个线程要做一个操作，比如，读文件，这个线程就会等待磁盘响应，此时cpu核心会处于未充分利用的状态。这是GCD就会发现这个状态，然后创建另一个线程来填补这个资源浪费空缺。

现在，想想上面的程序发生了啥？主线程非常迅速地将任务不断放入global queue中。GCD以一个少量工作线程的状态开始，然后开始执行任务。这些任务执行了一些很轻量工作后，就开始等待磁盘资源，慢得不像话的磁盘资源。

我们别忘记磁盘资源的特性，除非你使用的是SSD或者牛逼的RAID，否则磁盘资源会在竞争的时候变得异常的慢。。

刚开始的四个任务很轻松地就同时访问到了磁盘资源，然后开始等待磁盘资源返回。这时GCD发现CPU开始空闲了，它继续增加工作线程。然后，这些线程执行更多的磁盘读取任务，然后GCD再创建更多的工资线程。。。

可能在某个时间文件读取任务有完成了了。现在，线程池中可不止有四个线程，相反，有成百上千个。。GCD又会尝试将工作线程减少（太多使用CPU资源的线程），但是减少线程是由条件的，GCD不可以将一个正在执行任务的线程杀掉，并且也不能将这样的任务暂停。它必须等待这个任务完成。所有这些情况都导致GCD无法减少工作线程数。

然后所有这上百个线程开始一个个完成了他们的磁盘读取工作。它们开始竞争CPU资源，当然CPU在处理竞争上比磁盘先进多了。问题在于，这些线程读完文件后开始编码这些图片，如果你有很多很多图片，那么你的内存将开始爆仓。。然后内存耗尽咋办？虚拟内存啊，虚拟内存是啥，磁盘资源啊。Oh shit! ~

然后进入了一个恶性循环，磁盘资源竞争导致更多的线程被创建，这些线程导致更多的内存使用，然后内存爆仓导致虚拟内存交换，直至GCD创建了系统规定的线程数上限（可能是512个），而这些线程又没法被杀掉或暂停。。。

这就是使用GCD时，要注意的。GCD能智能地根据CPU情况来调整工作线程数，但是它却无法监视其他类型的资源状况。如果你的任务牵涉大量IO或者其他会导致线程block的东西，你需要把握好这个问题。

修正

问题的根源来自于磁盘IO，然后导致恶性循环。解决了磁盘资源碰撞，就解决了这个问题。

- Scrum敏捷项目管理
- 入侵的艺术
- C#入门经典(第3版)
- Java (JDK 6)学习笔记

博文推荐 更多>>



nowpaper
Windows
Phone专家



himi
Android开
发专家

- 结合经验浅谈System Design Document
- Apache虚拟目录日志分割及发布
- 不想当将军的士兵不是好士兵
- 国学大师南怀瑾

最新热帖 更多>>

- 【JAVA EE企业级开发四步走完全攻略
- 通用WAP网站生成系统 (PowerDiamond)
- 八类大学毕业生求职时不受欢迎
- IT人为什么难以拿到高薪？【转帖】
- 想去外企上班

GCD的custom queue使得这个问题易于解决。Custom queue是串行的。如果我们创建一个custom queue然后将所有的文件读写任务放入这个队列，磁盘资源的同时访问数会大大降低，资源访问碰撞就避免了。

虾米是我们修正后的代码，使用IO queue（也就是我们创建的custom queue专门用来读写磁盘）：

```
1.  dispatch_queue_t globalQueue = dispatch_get_global_queue(0, 0);
2.  dispatch_queue_t ioQueue = dispatch_queue_create("com.mikeash.imagegcd.io",
    NULL);
3.  dispatch_group_t group = dispatch_group_create();
4.  __block uint32_t count = -1;
5.  for (NSString *path in enumerator)
6.  {
7.      if ([[path pathExtension] lowercaseString] isEqual: @"jpg")
8.      {
9.          NSString *fullPath = [dir stringByAppendingPathComponent: path];
10.
11.          dispatch_group_async(group, ioQueue, BlockWithAutoreleasePool(^{
12.              NSData *data = [NSData dataWithContentsOfFile: fullPath];
13.              if (data)
14.                  dispatch_group_async(group, globalQueue, BlockWithAutorelea
sePool(^{
15.                      NSData *thumbnailData = ThumbnailDataForData(data);
16.                      if (thumbnailData)
17.                      {
18.                          NSString *thumbnailName = [NSString stringWithForma
t: @"%d.jpg",
19.                              OSAtomicIncrement32(&cou
nt;));
20.                          NSString *thumbnailPath = [destination stringByAppe
ndingPathComponent: thumbnailName];
21.                          dispatch_group_async(group, ioQueue, BlockWithAutor
eleasePool(^{
22.                              [thumbnailData writeToFile: thumbnailPath atomi
cally: NO];
23.                              }));
24.                          }
25.                      }));
26.                  }));
27.              }
28.          }
29.      dispatch_group_wait(group, DISPATCH_TIME_FOREVER);
```

这个就是我们的 imagegcd3.m.

GCD使得我们很容易就将任务的不同部分放入相同的队列中去（简单地嵌套一下dispatch）。这次我们的程序将会表现地很好。。我是说多数情况。。。

问题在于任务中的不同部分不是同步的，导致了整个程序的不稳定。我们的新程序的整个流程如下：

Main Thread	IO Queue	Concurrent Queue
find paths	-----> read	-----> process
		...
	write <-----	process

图中的箭头是非阻塞的，并且会简单地将内存中的对象进行缓冲。

现在假设一个机器的磁盘足够快，快到比CPU处理任务（也就是图片处理）要快。其实不难想象：虽然CPU的动作很快，但是它的工作更繁重，解码、压缩、编码。从磁盘读取的数据开始填满IO queue，数据会占用内存，很可能越占越多（如果你的~/Pictures中有很多很多图片的话）。

然后你就会内存爆仓，然后开始虚拟内存交换。。。又来了。。

这就会像第一次一样导致恶性循环。一旦任何东西导致工作线程阻塞，GCD就会创建更多的线程，这个线程执行的任务又会占用内存（从磁盘读取的数据），然后又开始交换内存。。

结果：这个程序要么就是运行地很顺畅，要么就是很低效。

注意如果磁盘速度比较慢的话，这个问题依旧会出现，因为缩略图会被缓冲在内存里，不过这个问题导致的低效比较不容易出现，因为缩略图占的内存少得多。

真正的修复

由于上一次我们的尝试出现的问题在于没有同步不同部分的操作，所以让我写出同步的代码。最简单的方法就是使用信号量来限制同时执行的任务数量。

那么，我们需要限制为多少呢？

显然我们需要根据CPU的核数来限制这个量，我们又想马儿好又想马儿不吃草，我们就设置为cpu核数的两倍吧。不过这里只是简单地这样处理，GCD的作用之一就是让我们不用关心操作系统的内部信息（比如cpu数），现在又来读取cpu核数，确实不太妙。也许我们在实际应用中，可以根据其他需求来定义这个限制量。

现在我们的主循环代码就是这样了：

```
1.  dispatch_queue_t ioQueue = dispatch_queue_create("com.mikeash.imagegcd.io",
    NULL);
2.
3.  int cpuCount = [[NSProcessInfo processInfo] processorCount];
4.  dispatch_semaphore_t jobSemaphore = dispatch_semaphore_create(cpuCount * 2)
    ;
5.
6.  dispatch_group_t group = dispatch_group_create();
7.  __block uint32_t count = -1;
8.  for(NSString *path in enumerator)
9.  {
10.     WithAutoreleasePool(^{
11.         if([[path pathExtension] lowercaseString] isEqual:@"jpg"])
12.         {
13.             NSString *fullPath = [dir stringByAppendingPathComponent: path]
    ;
14.
15.             dispatch_semaphore_wait(jobSemaphore, DISPATCH_TIME_FOREVER);
16.
17.             dispatch_group_async(group, ioQueue, BlockWithAutoreleasePool(^
    {
18.                 NSData *data = [NSData dataWithContentsOfFile: fullPath];
19.                 dispatch_group_async(group, globalQueue, BlockWithAutorelea
    sePool(^{
20.                     NSData *thumbnailData = ThumbnailDataForData(data);
21.                     if(thumbnailData)
22.                     {
23.                         NSString *thumbnailName = [NSString stringWithForma
    t:@"%d.jpg",
24.                                     OSAtomicIncrement32(&cou
    nt;));
25.                         NSString *thumbnailPath = [destination stringByAppen
    dingPathComponent: thumbnailName];
26.                         dispatch_group_async(group, ioQueue, BlockWithAutor
    eleasePool(^{
27.                             [thumbnailData writeToFile: thumbnailPath atomi
    cally: NO];
28.                             dispatch_semaphore_signal(jobSemaphore);
29.                             }));
30.                         }
31.                     else
32.                         atch_semaphore_signal(jobSemaphore);
```



```
33.         }));  
34.         }));  
35.     }  
36.     });  
37. }  
38. dispatch_group_wait(group, DISPATCH_TIME_FOREVER);
```

最终我们写出了能平滑运行且快速处理的程序。

基准测试

我测试了一些运行时间，对7913张图片：

程序	处理时间（秒）
imagegcd1.m	984
imagegcd2.m	没运行，这个还是别运行了
imagegcd3.m	300
imagegcd4.m	279

注意，因为我比较懒。所以我在运行这些测试的时候，没有关闭电脑上的其他程序。。。严格的进行对照的话，实在是太蛋疼了。。

所以这个数值我们只是参考一下。

比较有意思的是，3和4的执行状况差不多，大概是因为我电脑有15g可用内存吧。。。内存比较小的话，这个imagegcd3应该跑的很吃力，因为我发现它使用最多的时候，占用了10g内存。而4的话，没有占多少内存。

结论

GCD是个比较范特西的技术，可以办到很多事儿，但是它不能为你办所有的事儿。所以，对于进行IO操作并且可能会使用大量内存的任务，我们必须仔细斟酌。当然，即使这样，GCD还是为我们提供了简单有效的方法来进行并发计算。

【编辑推荐】

- 1. [GCD介绍\(一\):基本概念和Dispatch Queue](#)
- 2. [GCD介绍\(二\):多核心的性能](#)
- 3. [GCD介绍\(三\):Dispatch Sources](#)
- 4. [GCD介绍\(四\):完结](#)
- 5. [GCD实战一:使用串行队列实现简单的预加载](#)

【责任编辑：milk TEL：（010）68476606】

原文：[GCD实战二:资源竞争](#)

[返回移动开发首页](#)



微信扫一扫
移动首页版

同时，您也可以在移动端浏览器上输入“www.51cto.com”随时随地浏览和分享最具价值的技术内容

优质技术内容尽收中

微博推荐



51CTO移动开
51CTO移
动开发频道



51CTO官方微
51cto官方
微博



51CTO技术博
51CTO技
术博客官方



51CTO技术社
51CTO技
术社区(ho



51CTO熊平
51CTO传
媒总裁熊平



小林51CTO
北京无忧创
想信息技术

一键关注

注册微博

分享到:

1

收藏 | 打印 | 复制



给力
(1票)



动心
(0票)



废话
(0票)



专业
(0票)



标题党
(0票)



路过
(0票)

0 条评论，0 人参与。

★ 0



我有话说...

使用社交帐号登录

发布前先点击左边的按钮登录

最新评论

还没有评论

友言?

关于 [iOS多线程](#) [GCD实战](#) [资源竞争](#) 的更多文章

iOS多线程编程指南（拓展篇）

iOS多线程编程指南(四)线程同步

iOS多线程编程指南(三)Run Loop

iOS多线程编程指南(二)线程管理

iOS多线程编程指南(一)关于多线程编程

全面掌握iOS多线程攻略



多线程是一个比较轻量级的方法来实现单个应用程序内多个代码执行[详细]

栏目热门

更多>>

100分的输家：一个146年历史的诺基亚为何4
李开复：移动互联网时代该如何创业
开源免费天气预报接口API以及全国所有地区

同期最新

更多>>

GCD实战一:使用串行队列实现简单的预加载
GCD介绍(四):完结
GCD介绍(三):Dispatch Sources

http://mobile.51cto.com/hot-403011.htm

第 7 页（共 8 页）

三星或将推“土豪金”Galaxy S4 死磕iPhone 5 微软收购诺基亚：移动开发者应该看到什么	GCD介绍(二):多核心的性能 移动互联网时代的运营商步子应该怎么迈
---	---------------------------------------

移动开发 频道导航	
平台	移动Web Android iOS Windows Phone
应用	移动应用 移动团队 应用商店 专题汇总 Phone Club
观察	业界观察 调查数据 移动信息化
Android	热点 资讯 基础 多媒体 数据库 设计 工具 编译

热点推荐				
				
Android开发应用详解	那些性感的让人尖叫的程序员	HTML5 下一代Web开发标准详解	高性能WEB开发应用指南	Ubuntu开源技术交流频道

热门标签:	windows频道	移动开发	云计算	objective-c	tp-link路由器设置图解	html5
-------	---------------------------	----------------------	---------------------	-----------------------------	--------------------------------	-----------------------



上百度公益开放平台 为您的公益组织免费推广吧!

51CTO旗下网站				
领先的IT技术网站 51CTO	领先的中文存储媒体 WatchStor	中国首个CIO网站 CIOage	中国首家数字医疗网站 HC3i	