## 自我介绍

各位面试官，下午好。我叫严健康，本科毕业于复旦大学微电子系，现在是复旦大学微电子系研二的学生，研究方向主要是计算机辅助设计和分布式计算。先主要介绍一下我的项目经历。我2014年在复旦微电子公司实习期间参与了FPGA的布局自动化软件的并行化开发。我是用MapReduce的思想来并行化模拟退火算法最后实现了很好的加速比。2015年我主要在实验室负责分布式流媒体实时处理集群的建设。这个项目主要是基于storm来实现对大量摄像头画面的捕捉和实时图像处理，我作为项目的负责人一方面负责项目的整体架构，同时我阅读storm的Clojure部分的源代码并进行改进使storm能够支持GPU的调度。

我目前对C++以及linux比较熟悉，实验室基本上所有的服务器和对应的配置都是我完成的，比如svn,redmine,实验室的网站，备份，wiki, ldap，mvn等。

我觉得我的自学能力较强，遇到问题能够潜下心来寻找解决办法，我曾经为了理解神经网络并看懂一篇会议论文的代码，我花了一周的时间看了四十多篇论文。同时我有较好的文档能力，并且坚持每天写工作日志，记录遇到的问题。

我这次应聘的是运营开发岗位，因为我对云平台架构建设和运营方面比较感兴趣，我非常希望能够加入腾讯公司。

我的自我介绍就是这些，接下来请您多多指教，谢谢！

## 项目：并行布局

我2014年在复旦微电子公司实习期间参与了FPGA的布局自动化软件的并行化开发。FPGA是可编程逻辑门阵列，里面是各种固定资源，资源之间的连接关系通过开关盒来确定。FPGA布局其实就是在资源约束情况下将电路不同的模块(比如运算模块，IO模块)映射到在FPGA上对应的资源中，这些模块之间有确定的连接关系，因此采用模拟退火算法进行解空间的搜索。

就是一开始进行随机的布局，然后不断的随机交换模块，接受并计算连线和面积代价，接受使代价变小的交换，同时以一定的概率接受使代价变大的交换，这样的话避免陷入局部最优解。

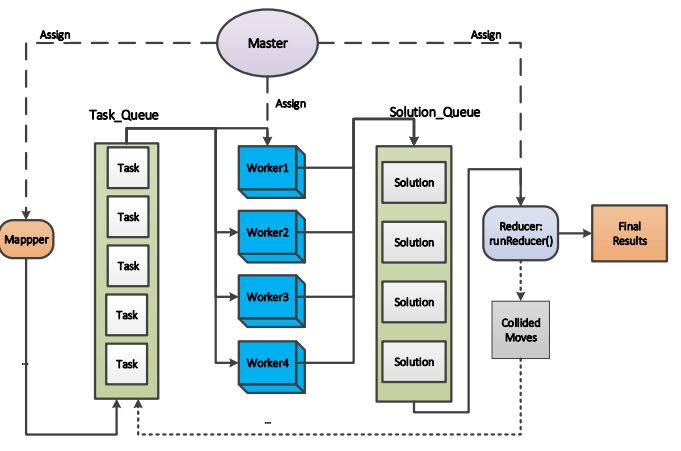
但是电路规模大的时候运行很慢，因此我主要是通过多线程并行的方式来提高布局速度。

运用到来的优化措施主要有：

1. 首先将一次模拟退火交换分为两个阶段，第一阶段占据时间80%，第二阶段20%时间。然后并行化第一阶段。
2. 为了保证串行等价性，没有采用默认的线性同余随机数发生器，采用了唐纳德在计算机程序艺术设计中提出的Coveyou随机数发生器，主要公式如下：



1. 使用的MapReduce框架：



首先，主线程在开始退火时，创建一个 Mapper 线程、若干 Worker 线程和一 个 Reducer 线程。Mapper 线程负责分配退火的任务，它将需要进行退火的 Move 放入任务队 列；Worker 线程负责执行每个 Move 的 Processing 阶段，它从任务队列中取出任 务，然后计算该次 Move 的位置和代价信息，并将计算结果放入结果队列中，然 后继续取出 Move；Reducer 从结果队列中取出 Move 执行完的结果， 然后先判断该 Move 是否与 发生冲突，如果不发生冲突的话，那么继续判断其能否被接受，如果被拒绝，那 么 Reducer 舍弃该次 Move 并继续取出下一次 Move，如果接受的话，那么 Reducer 要更新相应的数据信息；如果发生冲突的话，那么 Reducer 将此次 Move放回任 务队列重新处理。

1. 最大限度的减小重新执行的代价；
2. 在空间和时间方面做出权衡，使用内存池来减小new和delete的开销。通过共享数据结构来实现同步；
3. 通过编写python测试脚本进行自动测试。
4. 使用互斥锁和条件变量来实现子线程之间的同步和主线程和子线程之间的等待和唤醒。
5. 使用clock\_gettime来获取精确时间测试，通过关闭超线程和单独的测试机来进行测试。
6. 项目的复杂度在于任务之间的关联度很高，很难实现很高的并行；通过每次模拟退火至少要进行几百万次的迭代操作，因此多线程的调试比较复杂。

首先保证确定性，然后保证串行等价性，最后实现加速比。

MapReduce框架的代码实现：虚函数。

## 项目：分布式流媒体

2015年我主要在实验室负责分布式流媒体实时处理集群的建设。这个项目主要是基于storm来实现对大量摄像头画面的捕捉和实时处理，项目使用xuggle库从rtsp摄像头抓取视频流，解析为Frame之后存储到kafka消息队列中，然后storm再从kafka中读取存储的视频流进行实时的图像处理，处理完的结果会发送到一个RTMP的直播服务器，然后再通过前端的网页显示。Storm调用opencv和GPU进行图像处理，我作为项目的负责人一方面负责项目的整体控制，同时实现了项目的原型设计，同时我阅读storm的源代码并进行改进，加入了GPU的调度支持。然后这个项目已经完成了第一期的demo，接下来会进行GPU的调度优化和性能优化。

项目中遇到的问题：

1. ImageIO多线程read会出现阻塞，使用OpenCVIO；
2. 实验室的网络原因，Clojure的很多依赖包都无法下载下来，为了便于实验室的开发，搭建了内部的mvn仓库，仓库挂翻墙代理；
3. 一些动态链接库的依赖解决问题，包括路径问题和跨平台不兼容问题；
4. storm的动态调度；
5. storm源代码主要使用Clojure开发，函数式编程代码理解较困难；

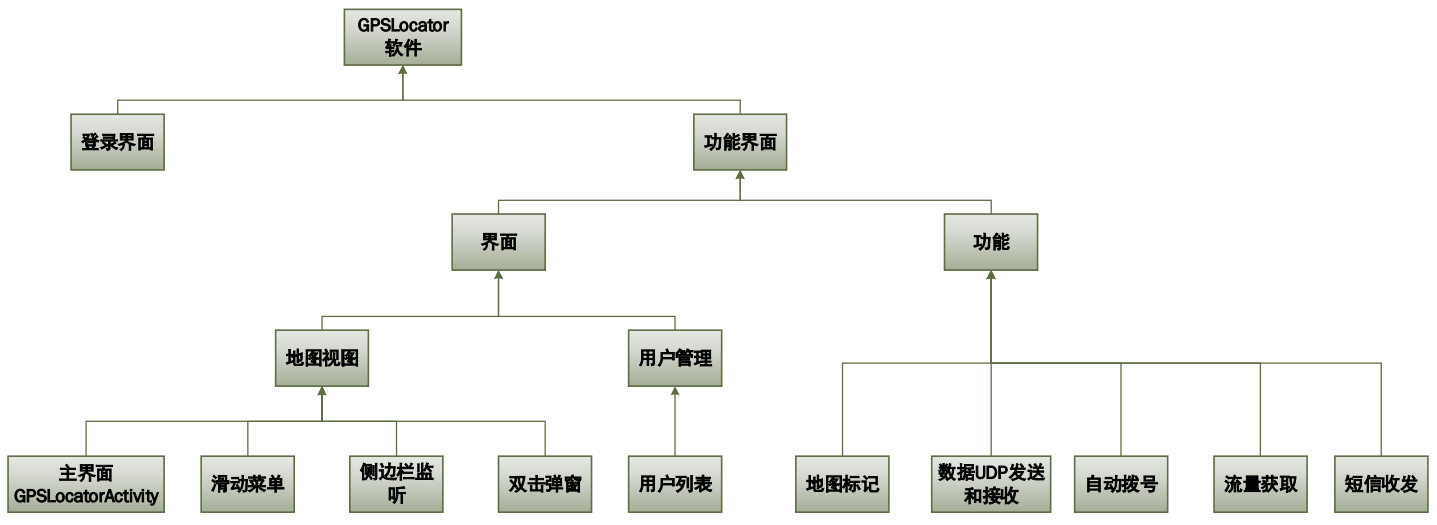
项目中用到的设计模式：

1. 简单工厂模式用来产生不同的文件Connector，比如ftp， http， local；根据location前缀来得到对应的文件Connector；
2. 桥接模式：定义基本的IOperation接口，然后分别实现single和batch，和opencv和GPU；

3）Façade模式：对用户提供一个统一的接口；

## 项目：移动定位项目

该项目是针对小孩老人进行位置监护的一个产品，家长、老师和管理人员可以通过使用自己的手机实时查询被监护人的位置。项目的开发主要包括硬件产品的研发，服务器后端信息存储和查询开发，前端网页界面开发和安卓客户端的开发，我主要负责安卓客户端的开发。

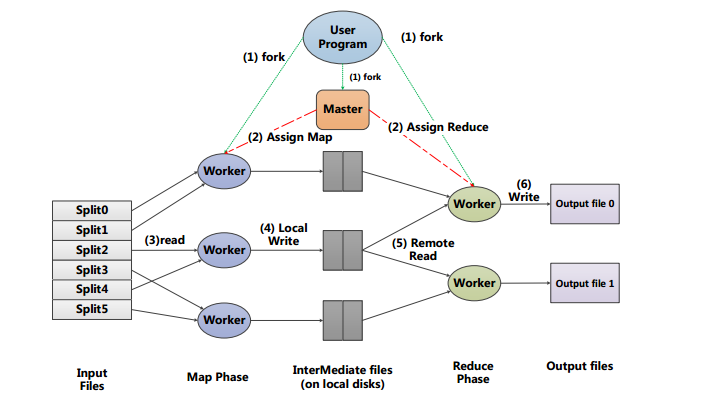


由于该项目规模较小，因此按照功能逻辑分为九个包：  
(1) Activity 包中是实现界面的所有类；  
(2) slideMenu 包中是实现滑动用户菜单的类；  
(3) data 包中是一些数据定义、存储和处理的类；  
(4) UDPHelper 包中是程序中在查询和同步时的 UDP 数据传输类；  
(5) mapMarker 包中是根据查询结果在地图上做标记的类；  
(6) autoCall 包中是一键呼叫被管理用户的类；  
(7) message 包中是关于短信寻呼时短信收发处理的类；  
(8) getbytes 包中是计算流量信息的类；



## MapReduce：

当用户调用 MapReduce 函数时，将发生下面的一系列动作（下面的序号和图 4-2 中的序号一一对应）：

  
1.用户程序首先调用的 MapReduce 库将输入文件分成 M 个数据片度，每个 数据片段的大小一般从 16MB 到 64MB(可以通过可选的参数来控制每个数据片 段的大小)。然后用户程序在机群中创建大量的程序副本。   
2.这些程序副本中的有一个特殊的程序–master。副本中其它的程序都是worker 程序，由 master 分配任务。有 M 个 Map 任务和 R 个 Reduce 任务将被分 配， master 将一个 Map 任务或 Reduce 任务分配给一个空闲的 worker。  
3.被分配了 map 任务的 worker 程序读取相关的输入数据片段，从输入的数 据片段中解析出 key/value pair，然后把 key/value pair 传递给用户自定义的 Map 函数，由 Map 函数生成并输出的中间 key/value pair，并缓存在内存中。  
4.缓存中的 key/value pair 通过分区函数分成 R 个区域，之后周期性的写入到 本地磁盘上。缓存的 key/value pair 在本地磁盘上的存储位置将被回传给 master， 由 master 负责把这些存储位置再传送给 Reduce worker。  
5.当 Reduce worker 程序接收到 master 程序发来的数据存储位置信息后，使 用 RPC 从 Map worker 所在主机的磁盘上读取这些缓存数据。当 Reduce worker 读取了所有的中间数据后， 通过对 key 进行排序后使得具有相同 key 值的数据聚 合在一起。由于许多不同的 key 值会映射到相同的 Reduce 任务上，因此必须进 行排序。如果中间数据太大无法在内存中完成排序，那么就要在外部进行排序。  
6.Reduce worker 程序遍历排序后的中间数据，对于每一个唯一的中间 key 值，Reduce worker 程序将这个 key 值和它相关的中间 value 值的集合传递给用户自 定义的 Reduce 函数。 Reduce 函数的输出被追加到所属分区的输出文件。  
7.当所有的 Map 和 Reduce