ARTS第一周(2019.3.24)

1. **Algorithm**

1020. Partition Array Into Three Parts With Equal Sum

Easy 0 2 Favorite Share

Given an array `A` of integers, return `true` if and only if we can partition the array into three **non-empty** parts with equal sums.

Formally, we can partition the array if we can find indexes `i+1 < j` with `(A[0] + A[1] + ... + A[i] == A[i+1] + A[i+2] + ... + A[j-1] == A[j] + A[j-1] + ... + A[A.length - 1])`

Example 1:

```
Input: [0,2,1,-6,6,-7,9,1,2,0,1]
Output: true
Explanation: 0 + 2 + 1 = -6 + 6 - 7 + 9 + 1 = 2 + 0 + 1
```

Success Details >

Runtime: 68 ms, faster than 100.00% of C++ online submissions for Partition Array Into Three Parts With Equal Sum.

Memory Usage: 12.5 MB, less than 100.00% of C++ online submissions for Partition Array Into Three Parts With Equal Sum.

Next challenges:

Triangle | Subarray Product Less Than K | Add to Array-Form of Integer

Show off your acceptance:

Time Submitted	Status	Runtime	Memory	Language

```cpp
class Solution {
public:
    bool canThreePartsEqualSum(vector<int>& A) {

        int sum = accumulate(A.begin(), A.end(), 0);

        if (sum%3!=0) return false;

        int equalSum = sum/3;
        int tempSum=0;
        int numOfFound=0;
        int arrSize=A.size();
        int i;

        for (i=0; i<arrSize; i++) {
            tempSum+=A[i];
            if(tempSum==equalSum) {
                numOfFound++;
                if(numOfFound>=2) break;
                tempSum=0;
            }
        }

        if (numOfFound==2 && i<(arrSize-1)) return true;

        return false;
    }
};
```

2. Review

Microservices *by James Lewis, Martin Fowler（2014.3.25）*

微服务是一种架构方式，将一个软件应用设计成一组可以独立部署的服务，服务之间通过简单的通信机制进行通信。服务通常围绕业务构建，并且可以自动部署。

（1）与单体架构比较
单体架构（Monolithic Architecture）

- 模块的改动周期绑定在了一起
- 很难保持一个好的模块结构
- 扩展（Scaling）必须整体进行

微服务架构

- 各个服务可以独立部署和扩展
- 模块边界清晰

（2）微服务架构的常见特征

a. 组件化通过划分微服务实现
优点：独立部署，组件间结构更清晰直接
缺点：远程调用开销大

b. 微服务的组织结构和业务结构接近
（康威定律：一个组织设计出的系统结构往往和该组织的沟通结构趋同）

c. 开发围绕产品而不是项目进行，团队对某个产品的整个生命周期负责

d. 微服务间通信通过轻量级的通信机制

e. 分散管理：拆分后的微服务可以分别采用不同的技术实现

f. 分散的数据管理 （问题：数据一致性）

g. 构建、部署、运维的自动化技术

h. 要有充分的容错设计（监控、日志等等）

i. 服务分解常常是对现有系统设计的改进方式，分解原则是尽量保持模块更换和升级的独立性

（3）微服务是未来吗？（注意此文写于2014年）

- 包括Amazon在内的一些大公司在转向微服务，有一些积极的结果，但最终结果还需要几年时间才能明朗
- 组件的边界较难把握，一旦划分不当，微服务的重构更加不易
- 如果组件间的分工协作不清晰，会增加微服务之间的连接复杂性
- 团队技能水平影响转型结果

3. Tip

PXE(Pre-boot Execution Environment)是由Intel设计的协议， 它可以使计算机通过网络启动。

协议分为client和server两端，PXE client在网卡的ROM中，当计算机引导时，BIOS把PXE client调入内存执行，并显示出命令菜单，经用户选择后，PXE client将放置在远端的操作系统通过网络下载到本地运

行。

bootstrap配置

bootstrap文件pxelinux.0在执行过程中，要读配置文件，所有的配置文件都放在/tftpboot/pxelinux.cfg/目录下。由于PXElinux具有为不同的PXE Client提供不同的Linux内核以及根文件系统的功能，所以要通过不同的配置文件名来区分出不同的PXE Client的需求。比如一个PXE Client由DHCP Server分配的IP地址为192.168.0.22，那么相对应的配置文件名为/tftpboot/pxelinux.cfg/C0A80016（注：C0A80016为IP地址192.168.0.22的十六进制表示）。如果找不到，就按照顺序C0A80016->; C0A8001->; C0A800->; C0A80->; C0A8->; C0A->; C0->; C->;default查找配置文件。

cat 0A6F0A01
PXE for 10.111.10.1
default ks
prompt 1
timeout 5
label ks
kernel pxeboot/vmlinuz
ipappend 2 append ks=http://10.111.10.1:8080/5.10.32.0/install/ks.tcl ksdevice=bootif root=/dev/ram rw initrd=pxeboot/initrd.img cmdline reboot=t console=ttyS2,115200n81

4. Share

负载均衡产品/方案比较

(1) NGINX

第四层： 支持UDP,TCP
第七层： 支持HTTP,HTTPS,Email
会话持久性：通过iptables提供支持，第三方模块 nginx-sticky-module 提供对cookies的支持

均衡算法：

- List itemround-robin (weighted)
- least-connections (weighted)
- least-time (weighted)
- hash
- 可以通过第三方模块扩展

(2) LVS

第四层： 支持UDP,TCP,SCTP
第七层： 支持弱
会话持久性：LVS persistence可以实现将来自同一TCP/IP连接客户端的
所有请求定向到一个特定服务器上。

均衡算法：

- List itemround-robin (weighted)

- least-connections (weighted)
- least-time (weighted)
- hash
- shortest expected delay
- never queue scheduling