SPL Base

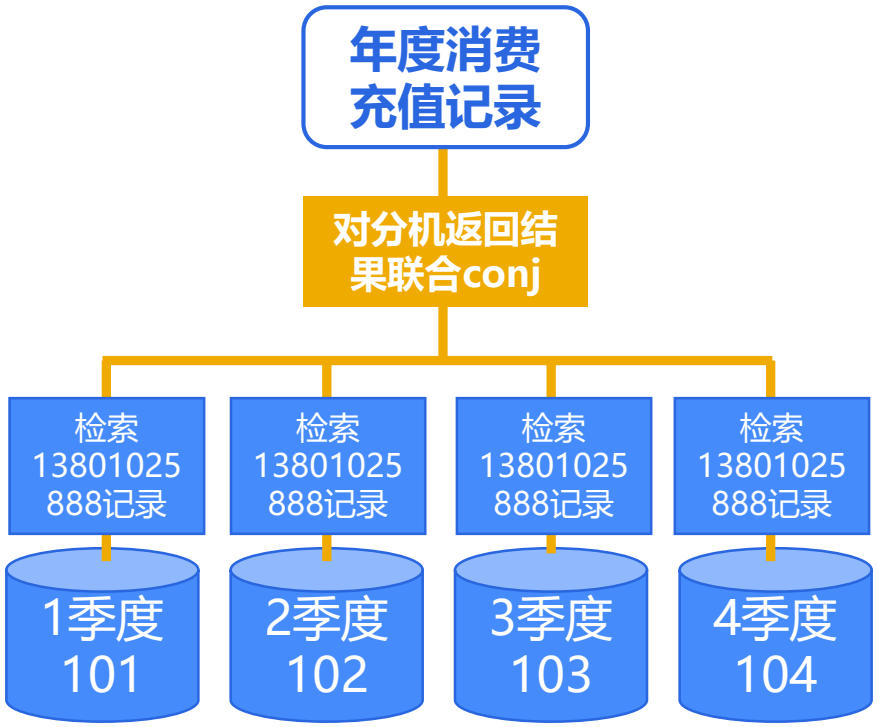
性能优化教案

集群

CONTENTS

多机查询—where

某通信公司用户查询年度消费充值记录



描述分机需要IP地址加上端口号，本文所有图示分机只取IP的后一节序号来代替。
比如分机101表示的是192.168.0.101:8281

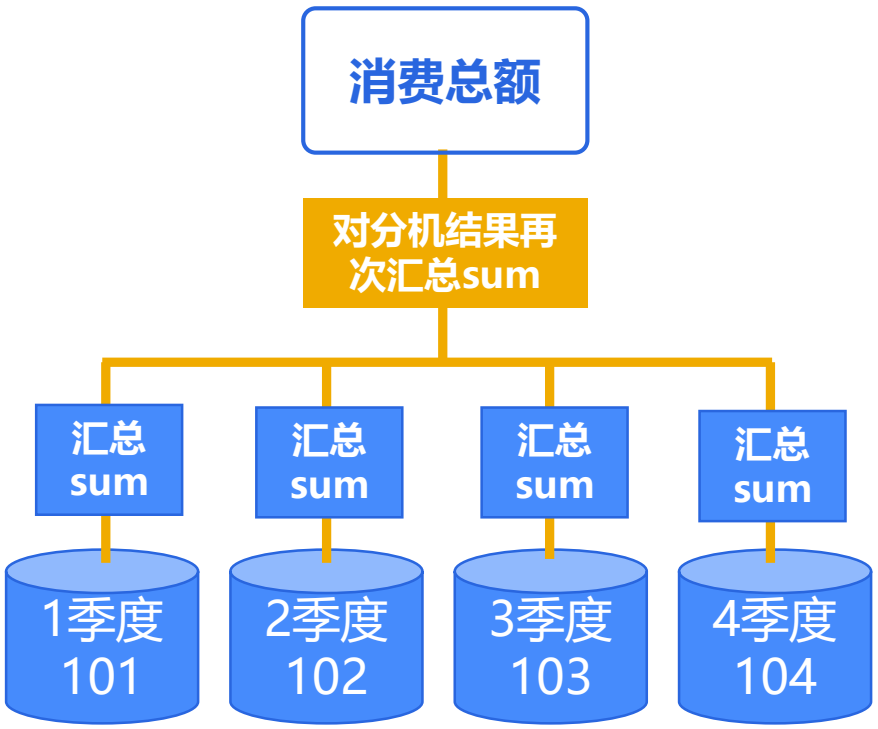
通信公司的消费记录表按季度拆分为4个文件，并分别保存到4台分机的“d:/data/consumer.ctx”下

fork语句将代码发送到各分机后并行执行，其中cursor里面的条件语句相当于where：

	A	B
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]	//4个季度数据分别存储在4台分机上
2	fork to(4);A1	=file("d:/data/consumer.ctx")
3		=B2.create().cursor(;mobile==“13801025888”).fetch()
4	=A2.conj()	//A4将各分机返回的结果联合

多机查询—sum

统计用户年度消费总额

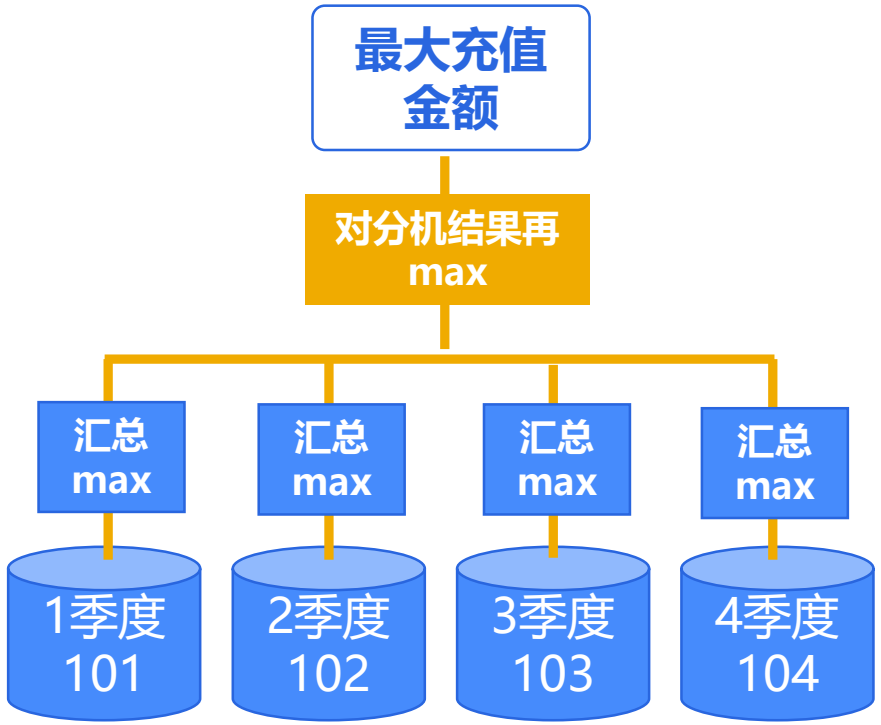


在分机上使用sum汇总，分机返回的结果需要再次sum：

	A	B
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]	//4个季度数据分别存储在4台分机上
2	fork to(4);A1	=file(“d:/data/consumer.ctx”).create()
3		=B2.cursor(;mobile==“13801025888”).total(sum(xfje))
4	=A2.sum()	//A4将各分机返回的结果再次汇总

多机查询—max、min

查询最大充值金额



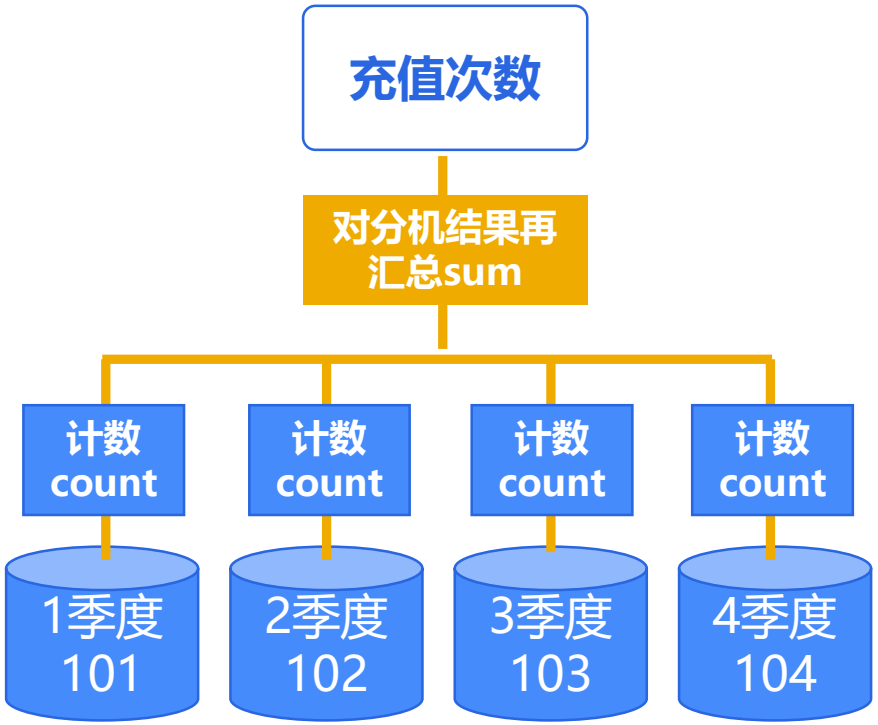
求最大值和最小值的方法类似，本案例给出最大值算法。

分机上使用max汇总，分机返回的结果需要再次max：

	A	B
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]	//4个季度数据分别存储在4台分机上
2	fork to(4);A1	=file(“d:/data/consumer.ctx”).create()
3		=B2.cursor(;mobile==“13801025888”).total(max(xfje))
4	=A2.max()	//A4将各分机返回的结果再次汇总

多机查询—count

统计充值次数

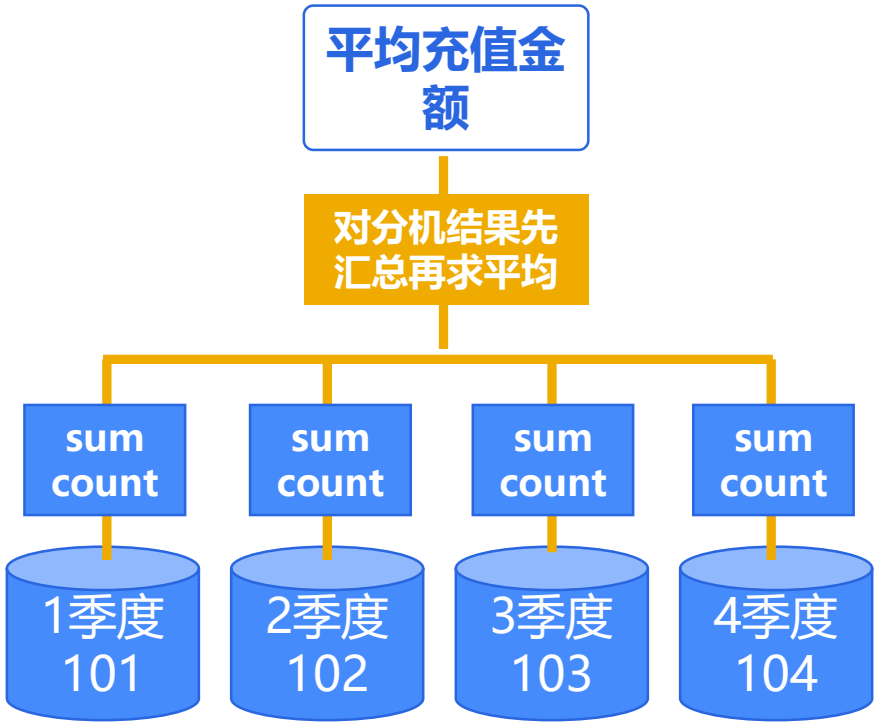


分机上使用count计数，分机返回的结果需要sum为总计：

	A	B
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]	//4个季度数据分别存储在4台分机上
2	fork to(4);A1	=file(“d:/data/consumer.ctx”).create()
3		=B2.cursor(;mobile==“13801025888”).total(count(~))
4	=A2.sum()	//A4将各分机返回的结果再次汇总

多机查询—avg

统计平均充值金额

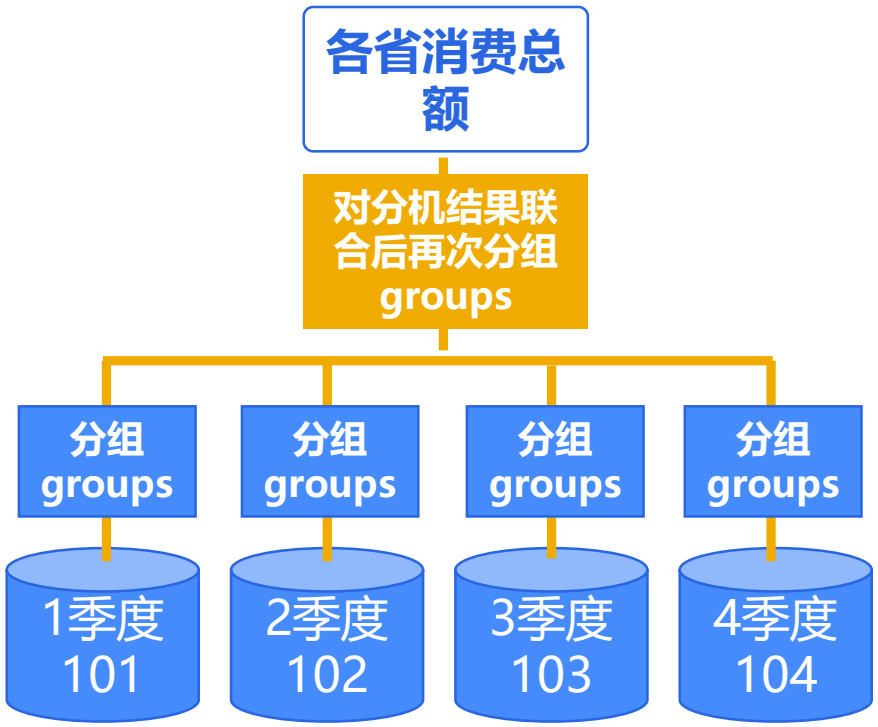


多机聚合运算中，avg不能针对分机返回的avg值再做avg。而要在分机上分别算出sum和count，然后在返回结果后算出总的sum和count，再算平均值：

	A	B
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]	//4个季度数据分别存储在4台分机上
2	fork to(4);A1	=file(“d:/data/consumer.ctx”).create()
3		=B2.cursor(;mobile== “13801025888”).group(mobile;sum(xfje):XFZE,count(~):COUNT)
4	=A2.conjx().total(sum(XFZE),sum(COUNT))	=A4(1)/A4(2)

多机查询—group by

统计年度各省消费总额

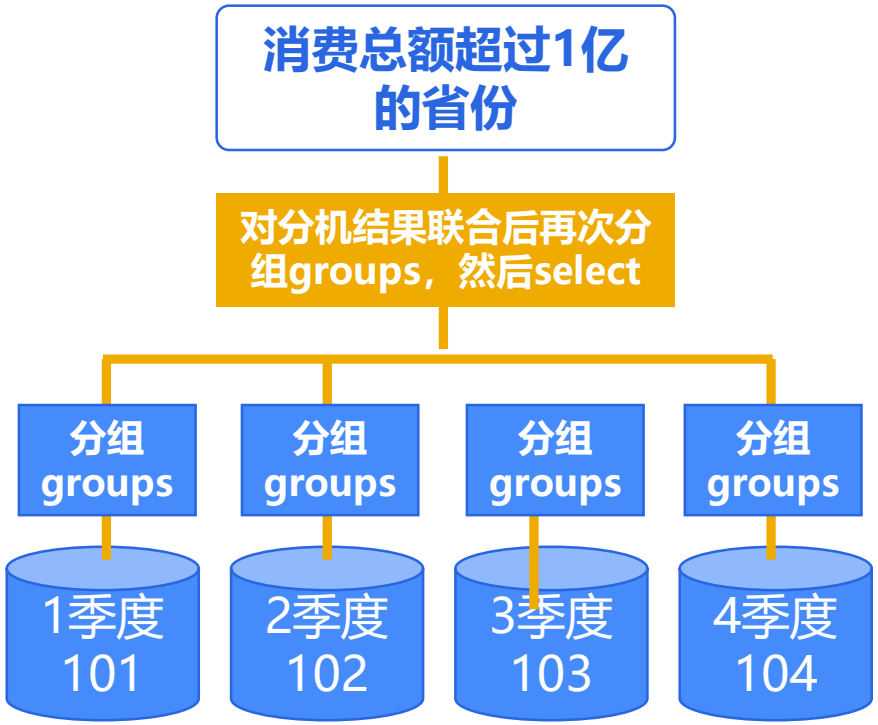


省的数量很少，分机分组后的结果很小，当然返回结果集后再分组依然很小。可以使用小分组方法groups返回结果序表，然后联合序表后再groups：

	A	B
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]	//4个季度数据分别存储在4台分机上
2	fork to(4);A1	=file("d:/data/consumer.ctx")
3		=B2.create().cursor().groups(province;sum(xfje):XFZE)
4	=A2.conj().groups(province;sum(XFZE):XFZE)	//A4将各分机返回的结果联合后再次分组

多机查询—having

统计消费总额超过1亿的省份

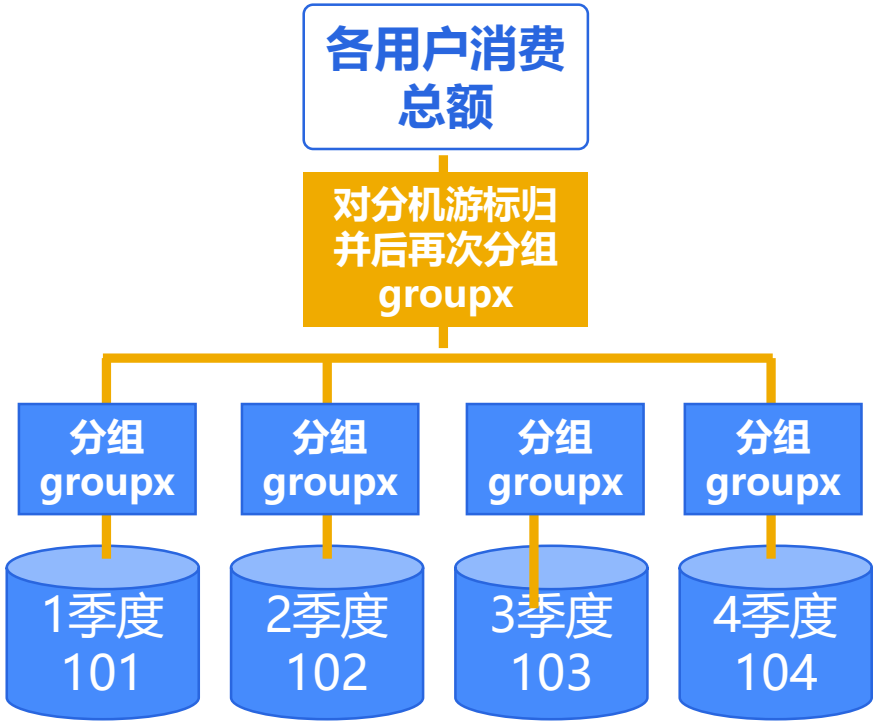


Having要在分组汇总之后再选出符合条件的记录。返回结果联合为总表后，再分组，然后选出符合条件的记录：

	A	B
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]	//4个季度数据分别存储在4台分机上
2	fork to(4);A1	=file("d:/data/consumer.ctx")
3		=B2.create().cursor().groups(province;sum(xfje):XFZE)
4	=A2.conj().groups(province;sum(XFZE):XFZE).select(XFZE>100000000)	//A4将各分机返回的结果联合后再次分组

多机查询—group by返回游标

统计各用户消费总额

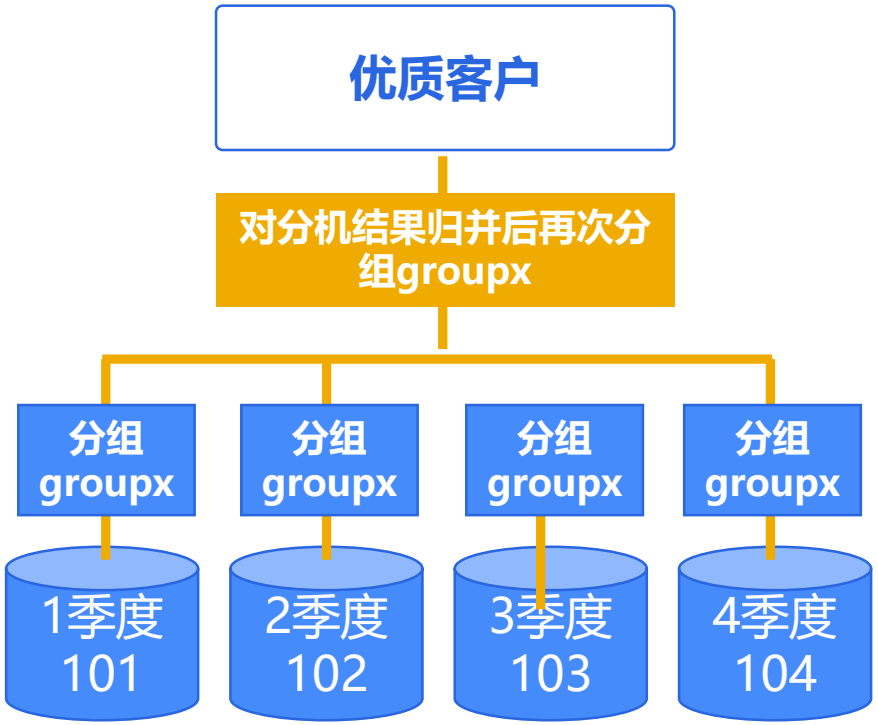


用户数很多，分组结果集太大，需要使用大分组方法groupx返回游标，其返回结果已有序，可以使用更高效的有序归并mergex和有序分组groupx@o实现最后汇总

	A	B
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]	//4个季度数据分别存储在4台分机上
2	fork to(4);A1	=file("d:/data/consumer.ctx") .create().cursor()
3		=B2.groupx(mobile;sum(xfje):XFZE)
4	=A2.mergex(mobile).groupx@o(mobile;sum(XFZE):XFZE).fetch(1000)	//A4将各分机返回的结果联合后再次分组

多机查询—distinct

统计单笔充值超过1000的客户



消费表中选出消费金额大于1000的记录后，由于存在手机号重复的记录，故需要distinct mobile，用无汇总项的groupx来实现：

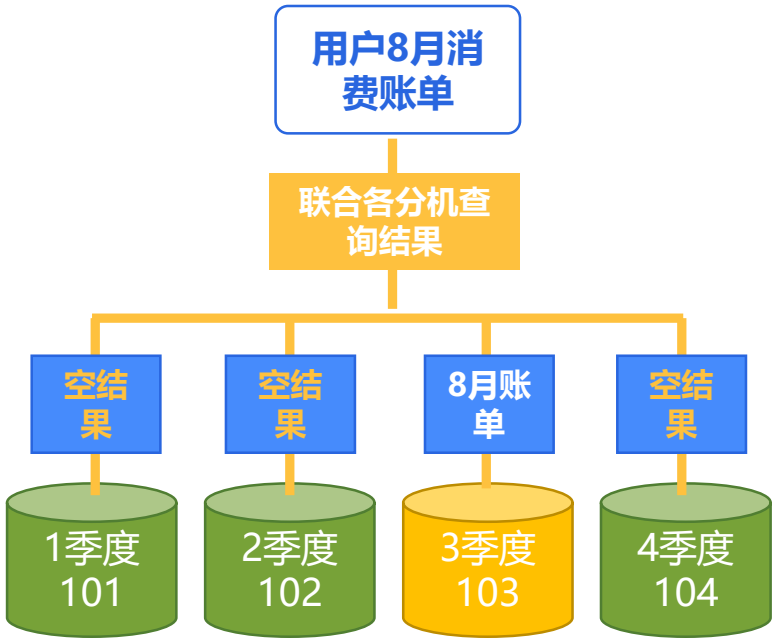
	A	B
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]	//4个季度数据分别存储在4台分机上
2	fork to(4);A1	=file("d:/data/consumer.ctx").create().cursor(;xfje>1000)
3		=B2.groupx(mobile)
4	=A2.mergex(mobile).groupx(mobile).fetch(1000)	//A4将各分机返回的结果联合后再次分组

CONTENTS

数据分布—季度分布的问题

查询某用户月消费账单

输入查询MOBILE=13801025888
查询月份MONTH=8
数据按季度分布时，我们发现查询8月账单时，数据全部集中在第3台分机，**没有负载均衡**



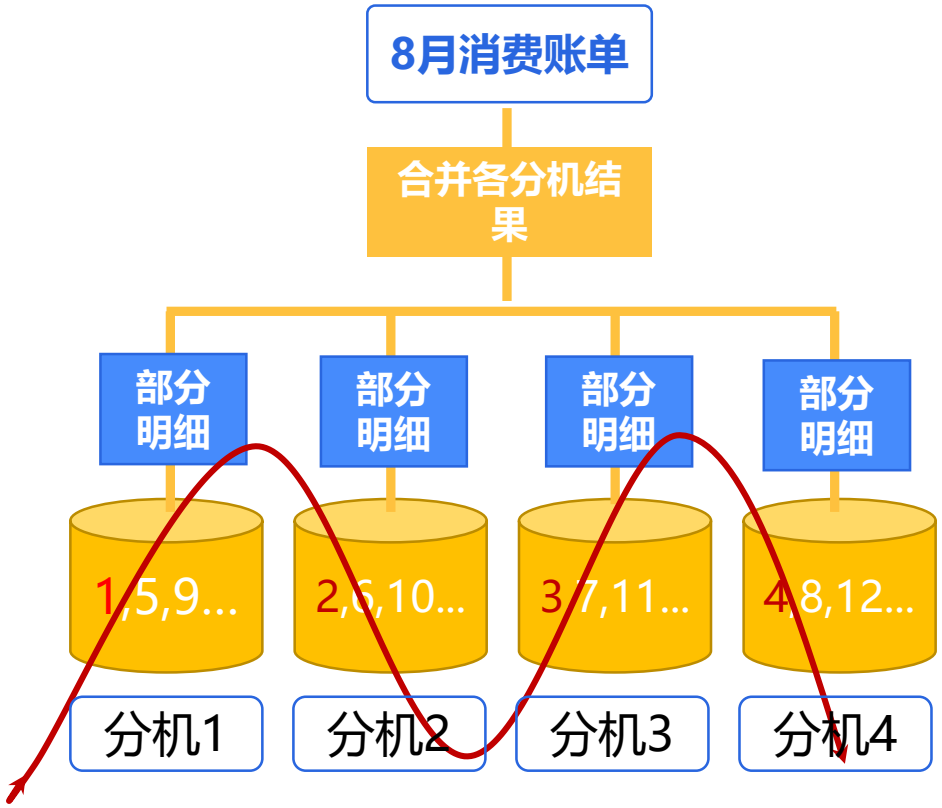
将查询参数MOBILE和MONTH置于fork语句中，发送到各分机去查询。

	A	B
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]	//4个季度数据分别存储在4台分机上
2	fork to(4),MOBILE,MONTH;A1	=file("d:/data/consumer.ctx").create()
3		=B2.cursor(;mobile==A2(2) && month(xfrq)==A2(3)).fetch()
4	=A2.conj()	

多个参数用逗号隔开

多个参数引用时需要指明序号

数据分布—蛇形分布



每台分机都有查询到数据，查询动作很好地做到了分机间负载均衡

季度拆分方式不合适将查询账单的任务均摊到每台分机。

如左图，可将记录按日期拆分，1号置于分机1，2号置于分机2，直至4号到分机4，然后5号又从分机1开始循环，日期n被分布到分机(n-1)%4+1。此时数据按照日期顺序像蛇一样分布于各台分机，故名蛇形分布。

此时返回值也是拆分的，要用merge做有序归并才能保证结果集的次序。

	A	B
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]	//4个季度数据分别存储在4台分机上
2	fork to(4),MOBILE,MONTH;A1	=file("d:/data/consumer.ctx").create()
3		=B2. cursor(;mobile==A2(2) &&month(xfrq)==A2(3)).fetch()
4	=A2.merge(xfrq)	

CONTENTS

集群聚合—概念

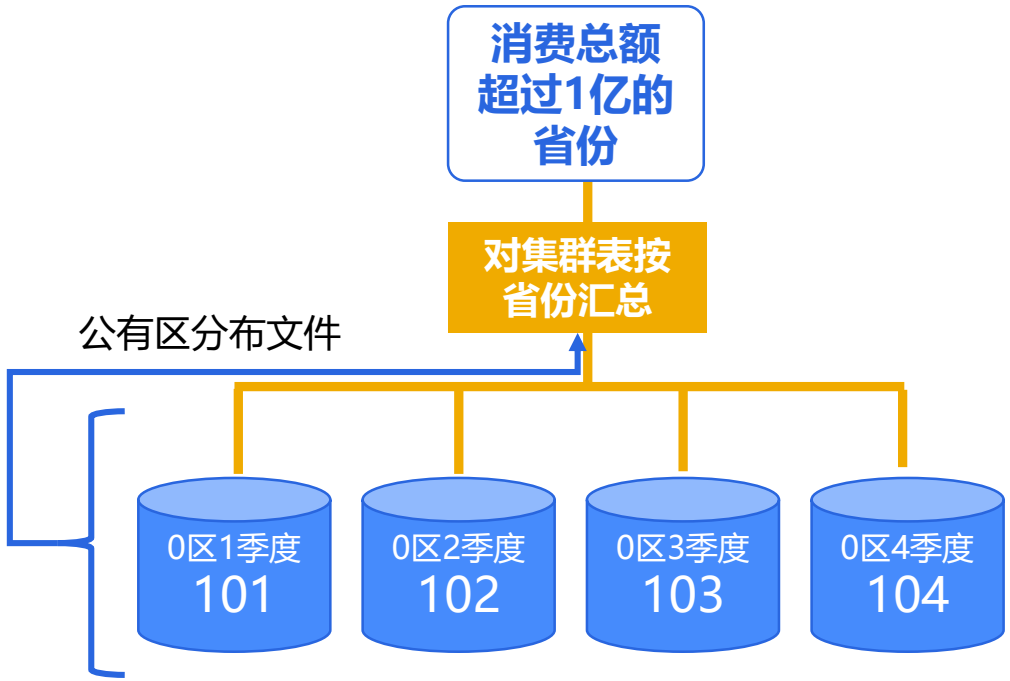
第一节中，我们将数据拆分到多机，利用集群提高计算性能。但同时也看到这些常见运算的细节仍较为繁琐，需要指明分机的执行动作才能得到正确的结果，明显比单机运算要复杂。

为了简化这些语法，我们基于集群建立逻辑表概念，使集群运算可以采用和单机运算类似的语法。集群表建立好后，程序员就不再关心分机的具体动作和相应的汇总处理，只需对集群表直接提出运算请求即可。

- **分区**　数据拆分成若干分区，简单地用自然数命名，分别存储于各分机
- **公有区**　0号区通常用于保存各分机都要使用到的维表，因此也叫公有区
- **保有区**　一台分机上可以有多个分区，这些分区统称为保有区
- **集群文件**　由多个分机上的数据联合构成的逻辑文件
- **集群表**　由集群文件产生的封装了并行计算的逻辑表
- **分布表**　各分机上的数据不同的集群表
- **复写表**　各分机上的数据相同的集群表

集群聚合—集群表理解

统计消费总额超过1亿的省份



我们使用集群表再来实现第一节中多机查询—having的算法：

	A
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]
2	=file@0z("consumer.ctx",A1)
3	=A2.create().cursor()
4	=A3.groups(province;sum(xfje):XFZE)
5	=A4.select(XFZE>100000000)

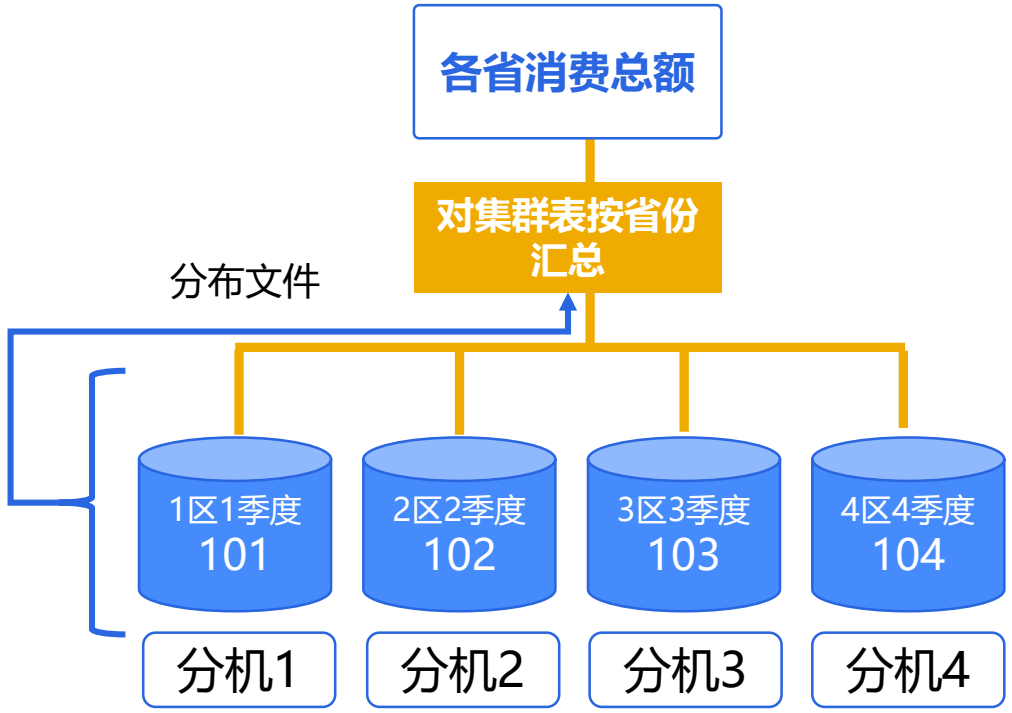
基于集群表的groups可以直接汇总出最终结果。相较于第一节中的算法，代码简单直接多了。

本例为对应第一节的文件分布，简单地将文件直接放在公有区，从而使用了0z选项，采用了公有区分布文件。实际应用中公有区常常用于公共维表存放，每个分机上数据相同，一般不会象本例中分机数据不同。

集群聚合—分布表

统计各省消费总额

z选项表示根据分机顺序，分别从分机1(192.168.0.101:8281)下加载1区，分机2下加载2区，依此类推。



分拆后的数据放在公有区不直观，难以知晓当前分机的数据是第几块，数据维护麻烦，所以分布表一般不存放在公有区。

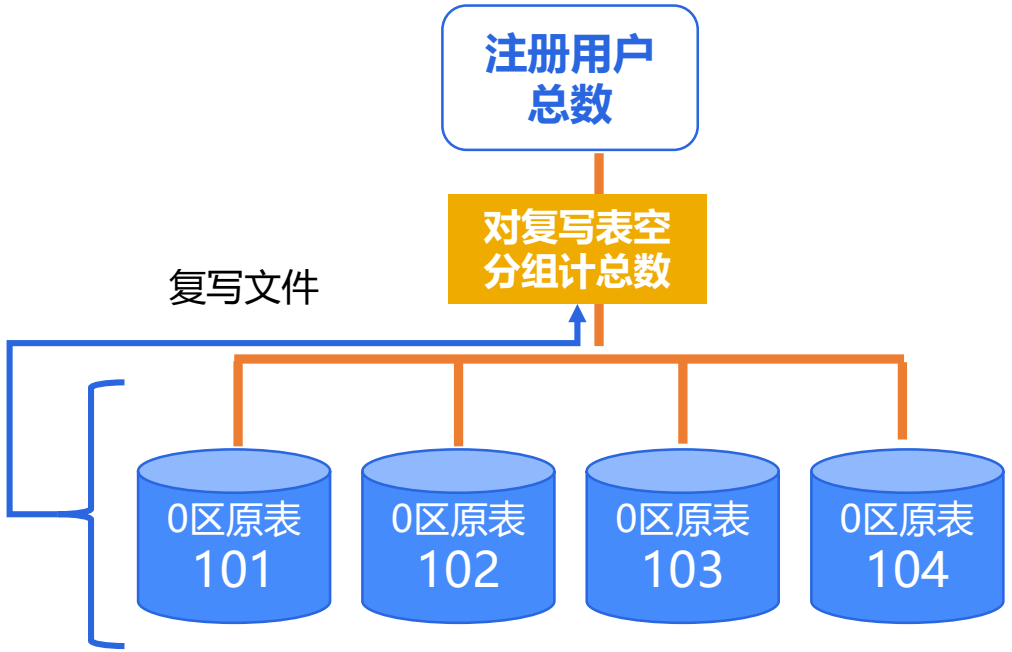
本例中的每块数据按拆分顺序，放在对应命名的分区号文件夹中。按季度拆分后，共有4个分区，每台分机对应一个区：

	A
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]
2	=file@z("consumer.ctx",A1)
3	=A2.create().cursor()
4	=A3.groups(province;sum(xfje):XFZE)
5	

集群聚合—复写表

统计注册用户总数

1、使用0选项，各分机读取相同的数据文件，产生复写文件



公共维表一般会放在公有区（即0区），每台分机都有相同的数据。对于这些被复制的数据，也可以利用集群来提高运算性能，即每个分机只处理其中一段数据。

	A
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]
2	=file@0("user.ctx",A1)
3	=A2.create().cursor@z()
4	=A3.groups(;count(~):COUNT)
5	//A3是集群表，直接对它做分组计数

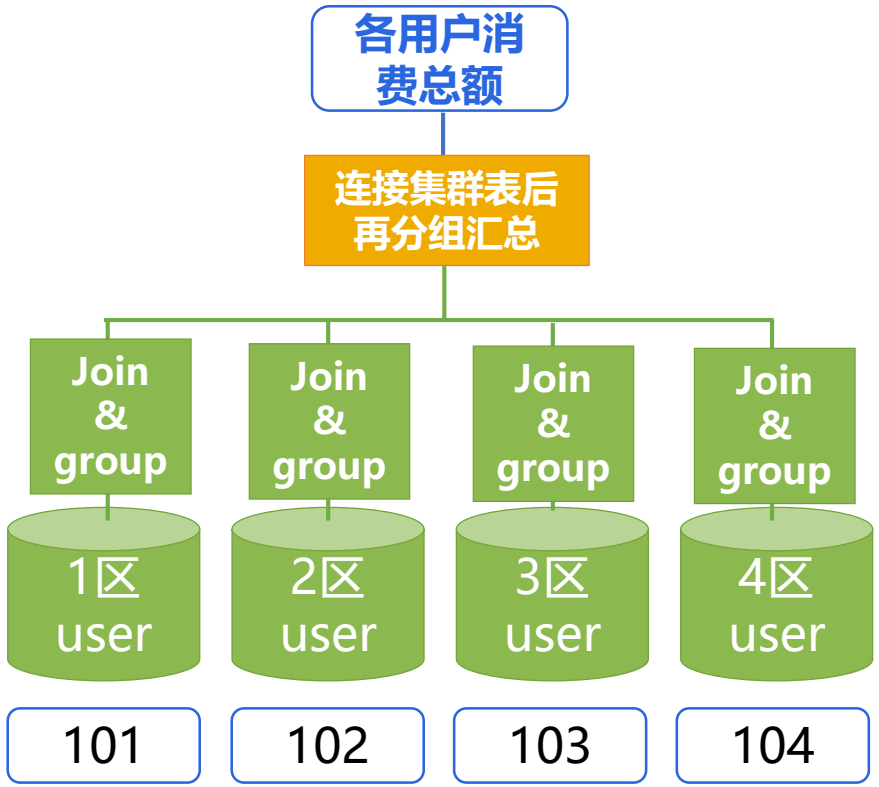
2、在复写表生成游标时，用z 选项自动拆分数据，每台分机加载一块，本例共4台分机，所以拆分为4块。

CONTENTS

连接运算—外键连接

统计各用户消费总额

1.消费表分成了4块，采用分布文件创建为分布表。



消费表仅记录手机号的缴费详情，要统计用户的消费总额，需要将手机号跟用户表user关联，取出用户名后再分组。

user表作为维表，放在每台分机上的公有区。使得分段后的消费表都能在分机内部完成连接：

	A	B
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]	
2	=file@z("consumer.ctx",A1)	=file@0("user.ctx",A1)
3	=A2.create().cursor()	=B2.create().memory()
4	=A3.join(mobile,B3,name)	
5	=A4.groupx(mobile:手机号,name:姓名;sum(xfje):消费总额).fetch(1000)	

2.用户表在每台分机的公有区都有一份拷贝，采用复写文件创建为复写表。

连接运算—外键连接-分机内连接图解

分机1连接后再分组

Consumer 1季度

id	mobile	xfrq	xfje
1	13610785578	2019-01-04	100
2	13710243564	2019-02-28	200
3	13801025999	2019-03-14	100
4	13901036854	2019-02-16	100
…			

user

mobile	name	province
13610785578	张三	北京
13710243564	李四	上海
13801025999	王五	广东
13901036854	刘蓓	湖南
…		

连接运算—外键连接-分段维表

统计各用户消费总额

跟消费表一样，也用分布文件创建分布表

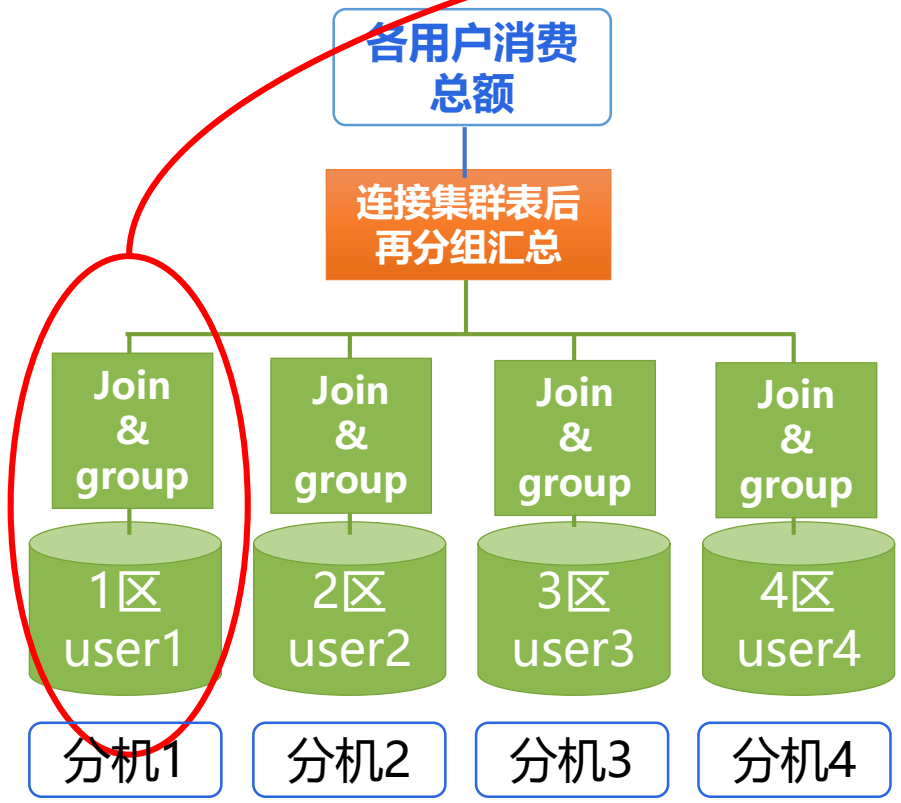
用户表(user)也可能非常大，无法全部装入一台分机的内存中。

此时可以将用户表也跟消费表一样拆分为4块，然后将每一块数据都分布到各台分机的对应分区。

	A	B
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]	
2	=file@z("consumer.ctx",A1)	=file@z("user.ctx",A1)
3	=A2.create().cursor()	=B2.create().memory()
4	=A3.join(mobile,B3,name)	
5	=A4.groupx(mobile:手机号,name:姓名;sum(xfje):消费总额).fetch(1000)	

连接运算—外键连接-分机间连接图解

分机1连接的user分布在4台分机



分机1的 1季度

id	mobile	xfrq	xfje
1	13610785578	2019-01-04	100
2	13710243564	2019-02-28	200
3	13801025999	2019-03-14	100
4	13901036854	2019-02-16	100
…			

分机1的user1

mobile	name	province
13610785578	张三	北京
13610785579	李辉	上海
13610785580	王五	广东
…		

分机2的User2

mobile	name	province
…		
13710243564	李四	上海
13710243565	刘明	福建
13710243566	谭海	湖南
…		

连接运算—主子表连接

某电商统计年度用户消费总额

主子表可能都很大，但不同于分段维表的连接。主子表或者同维表的连接，在有序分布后，均只需在分机内部连接以及分组，不必跨机访问。

对于主子表和同维表的连接，在保证数据都对主键有序时，可以使用有序归并的算法，不必象维表那样需要随机访问。在分布这类数据时，需要将数据同步分段，即关联记录分布在同一台分机。

本例统计每个用户的消费总额，只需将订单跟订单明细关联。将订单和对应订单明细表均按季度拆分后分布到4台分机。 分机在做连接时不涉及其它分机，减少网络传输。

	A	B
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]	//4个季度数据分别在4台分机
2	=file@z("订单.ctx",A1)	=file@z("订单明细.ctx",A1)
3	=A2.create().cursor()	=B2.create().cursor()
4	=joinx(A3:order;B3:ordermx, 订单ID).derive@x()	
5	=A4.groupx(客户ID:客户名;sum(数量*单价):消费金额).fetch(1000)	

连接运算—主子表连接-多路游标

统计年度用户消费总额

分机内部也在使用多路游标并行计算。

1、用m选项将订单产生为多路游标，游标路数为环境缺省。

分机上运算也可以象单机一样使用多线程并行计算。

类似地，分机上主子表并行计算时也要求数据同步分段。具体细节可参考单机上连接运算的相应内容

	A	B
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]	//4个季度数据分别在4台分机
2	=file@z("订单.ctx",A1)	=file@z("订单明细.ctx",A1)
3	=A2.create().cursor@m()	=B2.create().cursor(;;A3)
4	=joinx(A3:order;B3:ordermx, 订 单 ID).derive@x()	//B3生成跟A3路数相同且同步分段的游标
5	=A4.groupx(客户ID:客户名;sum(数量*单价):消费金额).fetch(1000)	

2、产生订单明细的游标时，得用参数A3，产生跟订单游标同步分段的游标。

CONTENTS

外存容错

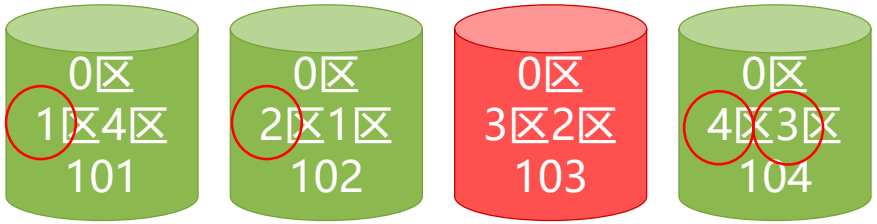
现在回过头再看下第3节中的集群表：

	A
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]
2	=file@z("订单.ctx",A1)

为保证某些分机出错时仍能不中断服务。我们可在每台分机上存储两个分区（如下图）

创建集群文件时，使用hosts函数找出一套包含所有分区的分机就行。

如下每台分机都放置了两个季度的内容，假设仍然是103出错时：



则*hosts(A1,4)*会找出图示红圈顺序的分机序列：

192.168.0.101:8281

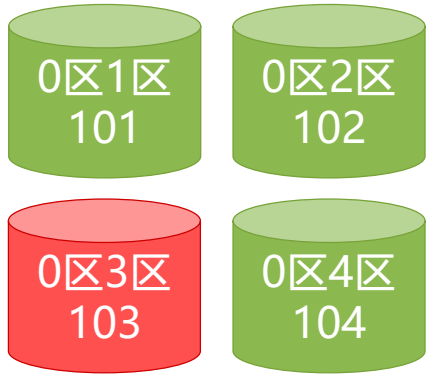
192.168.0.102:8281

192.168.0.104:8281

192.168.0.104:8281

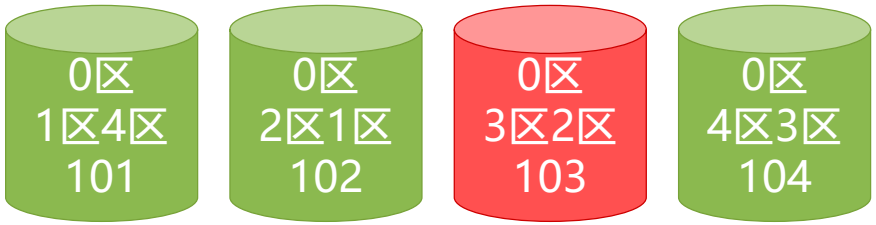
订单表被拆分成4个分区，每台分机有一个分区。

此时如果分机103出错，那么服务就无法继续了。其实还有3台分机可以工作，但数据已经不完整了。

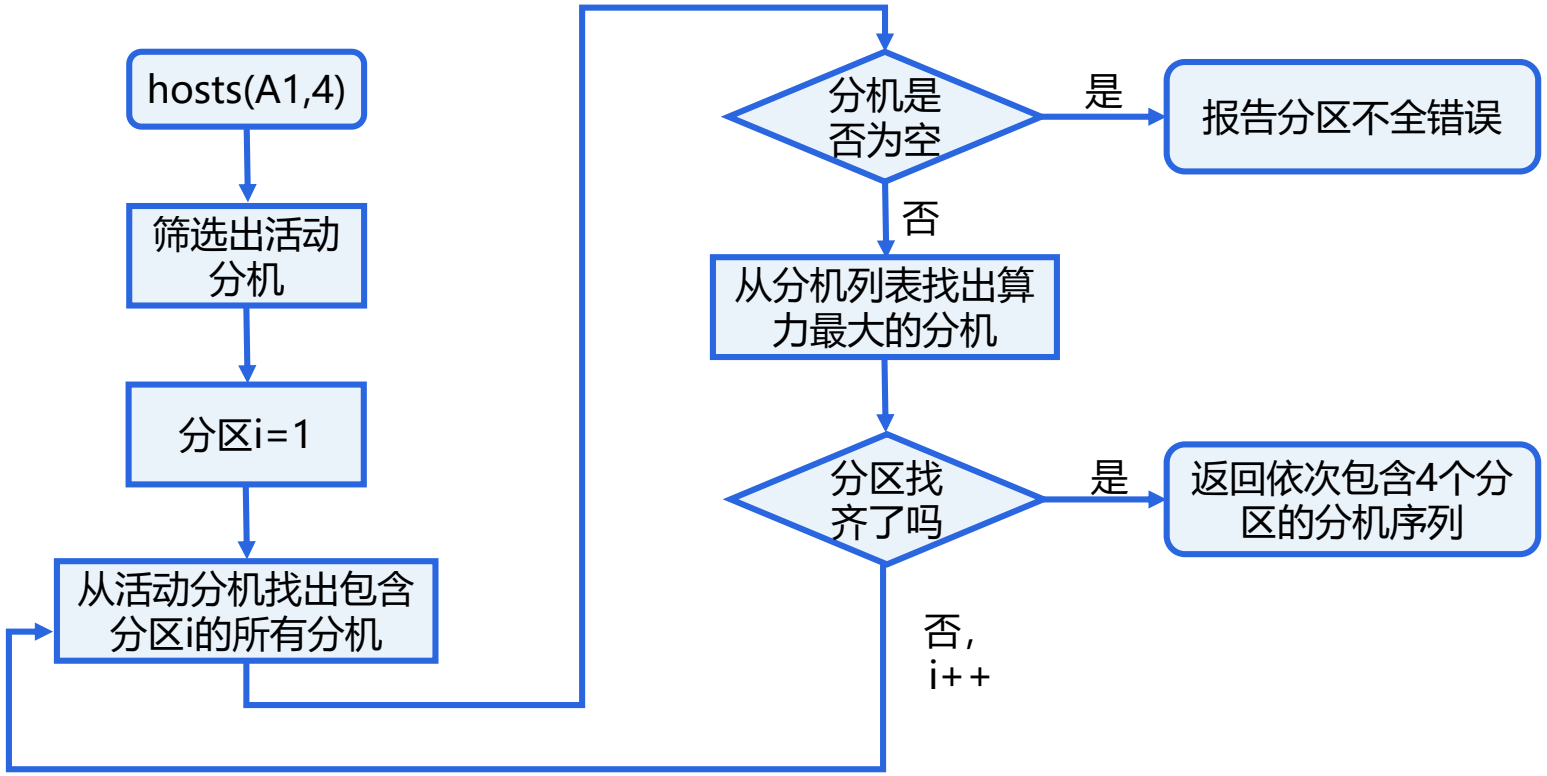


外存容错-hosts函数

如何根据分机的分区情况来找出相应分机序列呢？看下hosts的工作流程，A1网格定义了4台分机，其中分机103坏掉了（图中红色分机）：



让hosts(A1,4)从A1定义的4台分机中找出包含1到4区的分机列表：



外存容错-案例

统计年度用户消费频率

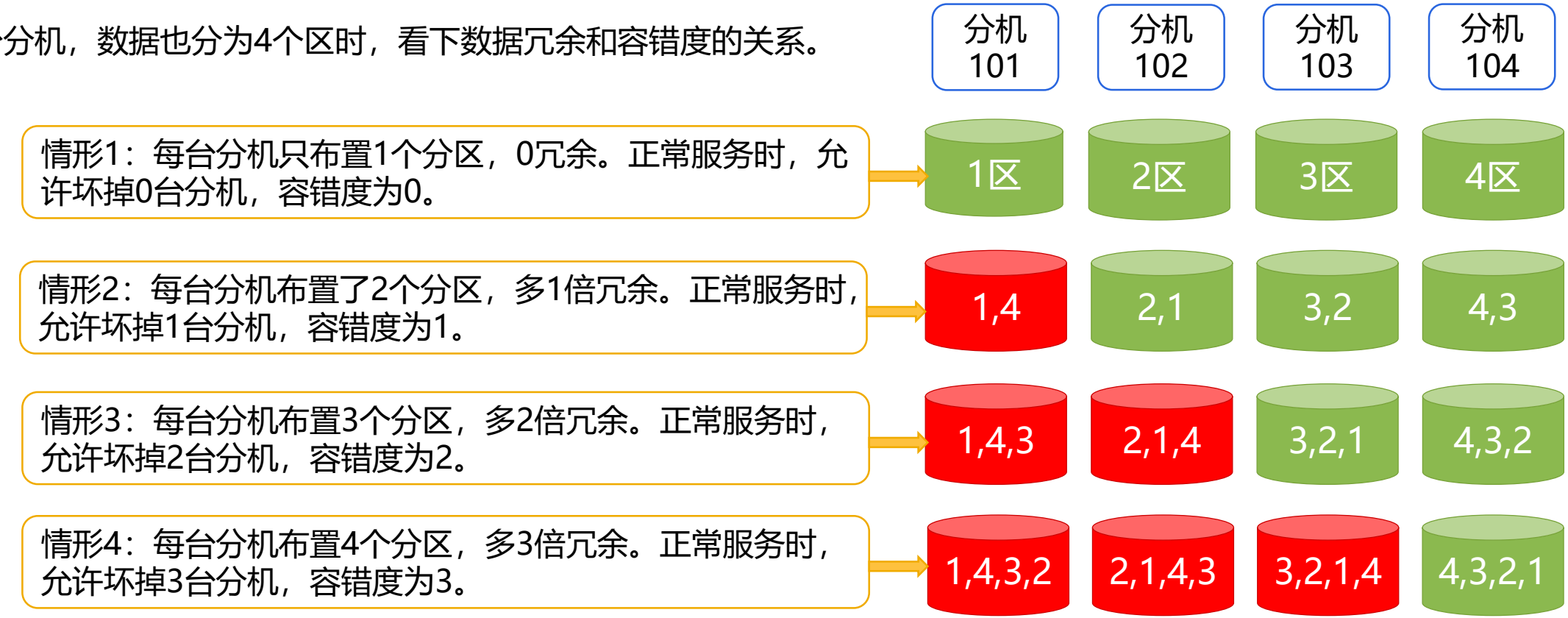
从上一节中可以知道，允许容错时，只要使用hosts函数找出可用分机序列，再将结果传给集群文件即可。

从4台包含冗余分区的分机列表中，依次找出包含4个季度（分区）的分机序列

	A
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]
2	=hosts(A1,4)
3	=file@z("consumer.ctx",A2)
4	=A3.create().cursor()
5	=A4.groups(;count(~):XFCS)

外存容错-容错度

现在从4台分机，数据也分为4个区时，看下数据冗余和容错度的关系。



由此可见，数据的冗余倍数，跟容错度是相等的。即数据分布为k倍冗余时，容错度也为k。

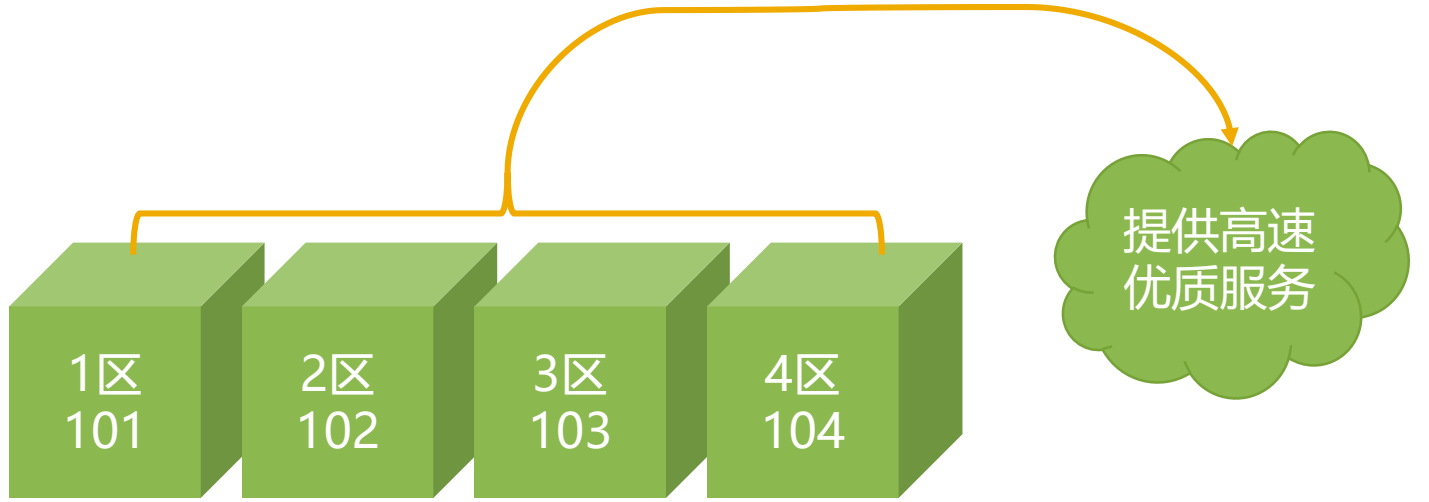
理论上，容错度看似越大越强壮，但同时数据的冗余倍数也越高，维护也越麻烦。所以实际生产环境中，要根据机器配置、业务中断容忍时间、数据量的多少和维护成本等，选择某个适合自己的容错度。

CONTENTS

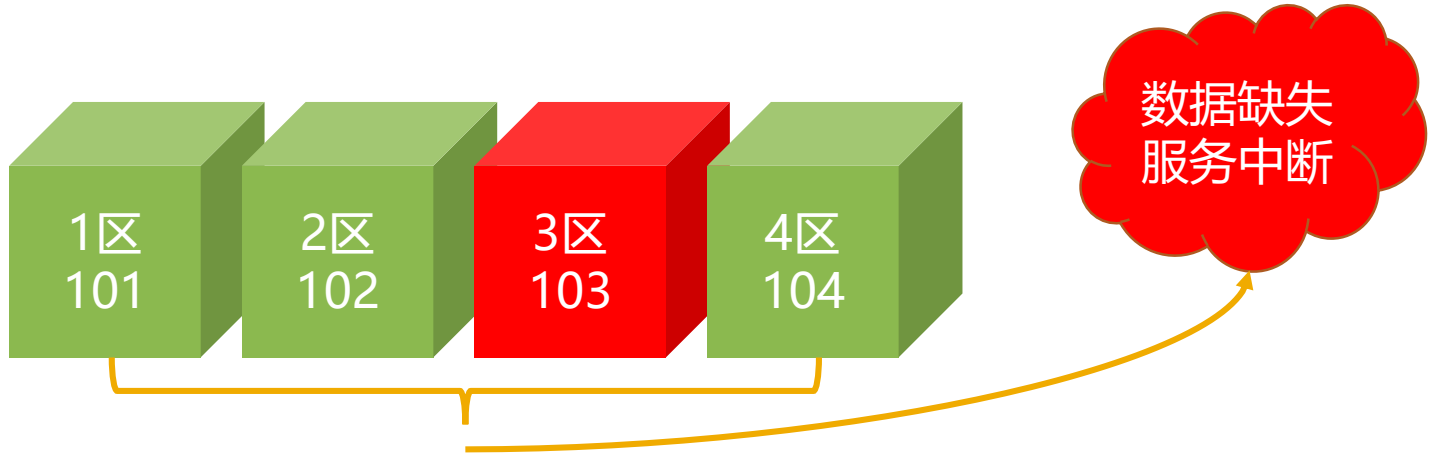
内存容错

为了提高性能，我们会把某些数据常驻内存（比如前面说的维表）。那么当机器故障时，这些内存数据也会失效，这时又如何提供持续的服务呢？可以采用外存相同的办法吗？

4台分机，每台分机常驻一部分数据



此时只要有一台分机坏掉，整个服务就崩掉了。显然需要有冗余数据，以增强系统的可用性。



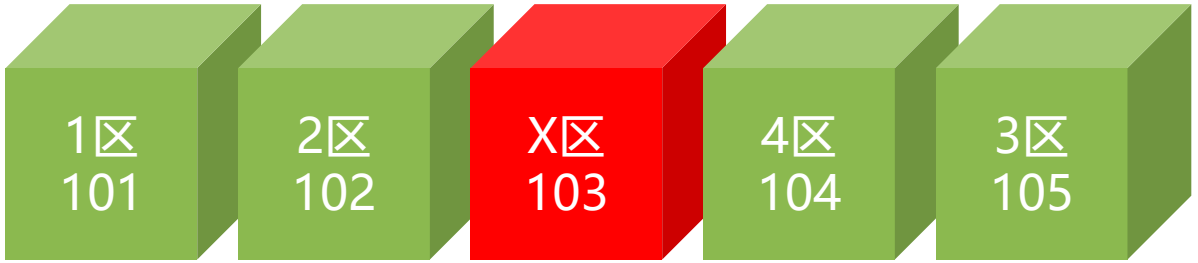
内存容错-备胎机

外存容错时，k容错度要求数据多复制k份，则存储利用率只有1/(1+k)。当容错度为1时，存储利用率只有50%。这对于外存是可以容忍的（硬盘几乎可以认为无穷大），但内存则不可接受。

内存容错时，我们将每台分机均满载一块数据，再留出几台用于容错的的**备胎机**。k容错度（允许k台机器失效）时，需要k个备机。提供服务的分机数为n时，内存利用率为n/(n+k) 。4台分机+1台备机可构成容错度为1的体系，内存利用率为4/(4+1)=80%，比同等情形下的外存容错机制的存储利用率高了很多。

现在是5台分机，只要找出4台分机并分别常驻1个季度的数据即可。当前图示找到前4台分机并分别加载了相应分区。105空闲备用

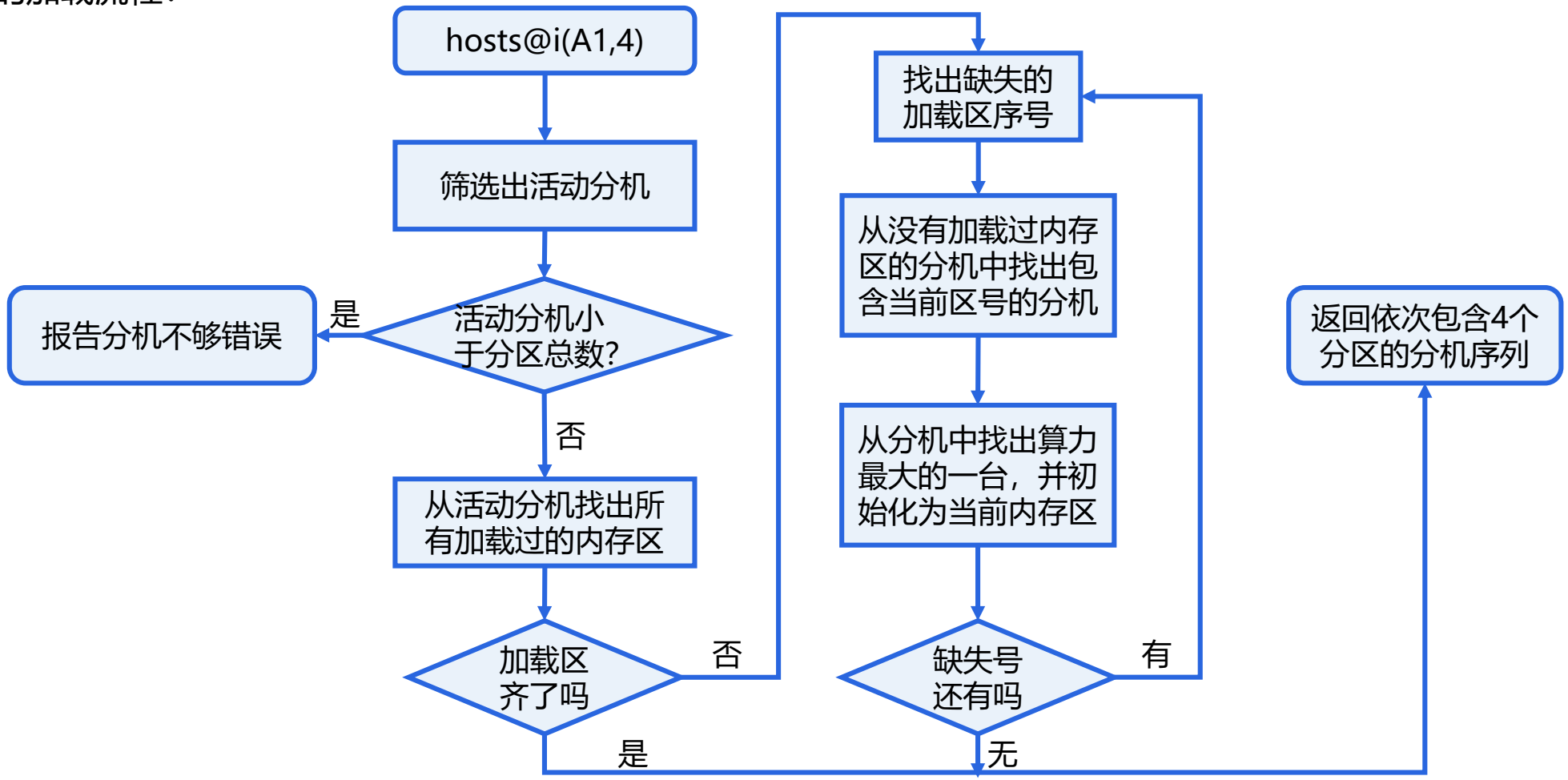
当103分机坏掉时，将105找出来，加载好3区数据，顶替坏掉的103



内存容错-hosts@i函数

hosts@i函数会自动复用已经加载过的内存区，内存区缺失时，自动找出备胎机并加载好相应的分区。

如下为hosts@i(A1,4)的加载流程：



内存容错—加载数据

分机加载的内存数据可能很多，通常需要写一段程序来实施，然后被host@i函数实时加载。

在加载程序中需要实现：

1、登记当前分机加载的区号，host@i函数要根据每台分机的区号来识别该分机是否已经加载过。

2、加载该分区数据，并设置全局变量，以供后续的计算任务引用。

一个加载数据的简单示例：

	A	B
1	//...	//z等于0时的公共初始化
2	if z>0	>zone(z)
3		=file("consumer.ctx":z)
4		=B3.create().cursor().fetch()
5		>env(SEASON,B4)

B2使用zone(z)将当前的加载区z登记到分机，防止下次host@i时重复加载。

B5用env(SEASON,B4)将B4中加载好的序表，设置到名为SEASON的全局变量，memory()函数即可用此变量创建集群表。

内存容错—案例

统计年度用户消费频率

跟外存容错的案例类似，在创建集群表之前，只要找出包含全部分区的分机序列即可。不同之处在于内存容错查找分机序列时，使用i选项。

host@i会保证列出包含指定分区的分机序列。

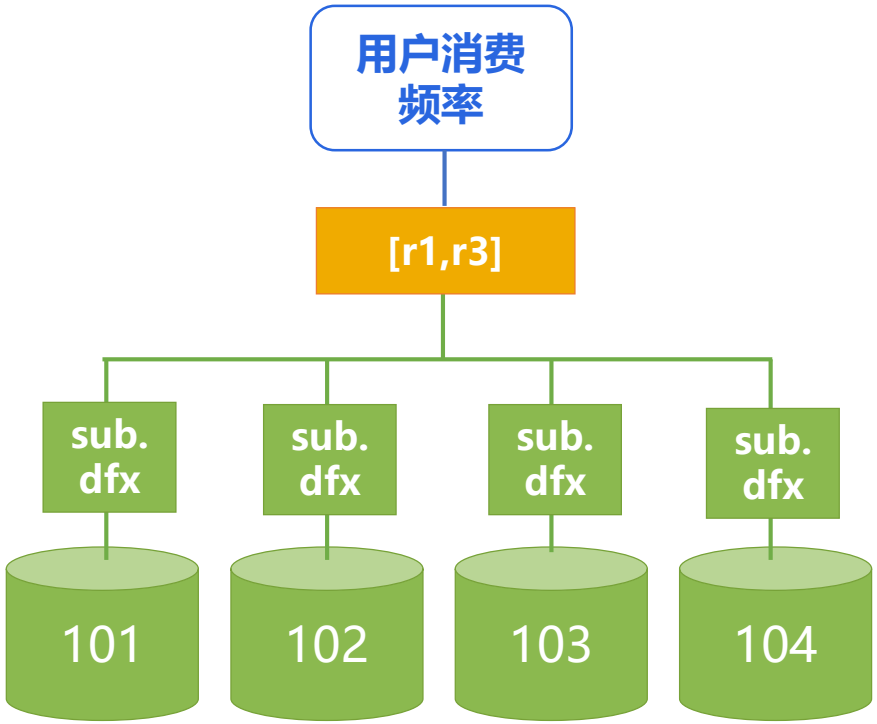
	A	B
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281","192.168.0.105:8281"]	//共准备5台分机，1台备用
2	=hosts@i(A1,4)	//从分机序列找出包含4个分区的分机
3	=memory(A2,SEASON)	//将各分机的内存数据(SEASON)构成一个集群表
4	=A3.groups(;count(~):XFCS)	//对该集群表进行次数统计

CONTENTS

任务分配—少量作业

当计算逻辑复杂到仅用集群表不足以描述时，就需要再写代码实现，这些代码被称为子程序。子程序部署到集群分机上，再由主控代码的callx函数来调用，一次调用过程被称为一个**作业**。作业计算完后，将结果组织成集合返回主机再作汇总处理。

一次大数据量的任务计算，通常会用参数拆分为多个作业去并行计算，以起到分机间的负载均衡。



设作业个数为n，分机台数为m。

- **少量作业** 当n<m时，我们称之为少量作业。

少量作业时，作业会被优先分配到最大算力的分机，如果最大算力的分机有多个时，再随机选一台。

	A
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]
2	=callx("sub.dfx",[1,3];A1)
3	=A2.sum()
4	//A2用了两个参数来调用子程序sub.dfx

任务分配—sub.dfx

作为示例，当前子程序sub.dfx仅定义了一个参数season，用来获取不同季节的数据文件。统计的逻辑也很简单，仅仅统计当前季度的消费次数：

	A
1	=file("consumer.ctx":season)
2	=A1.create().cursor()
3	=A2.groups(;count(~):COUNT).COUNT
4	return A3

callx会从指定分机中动态找出算力最大的分机来计算，所以具体由哪台分机来计算某个作业并不固定。这也是为了保证分机之间的负载均衡。

任务分配—等量作业

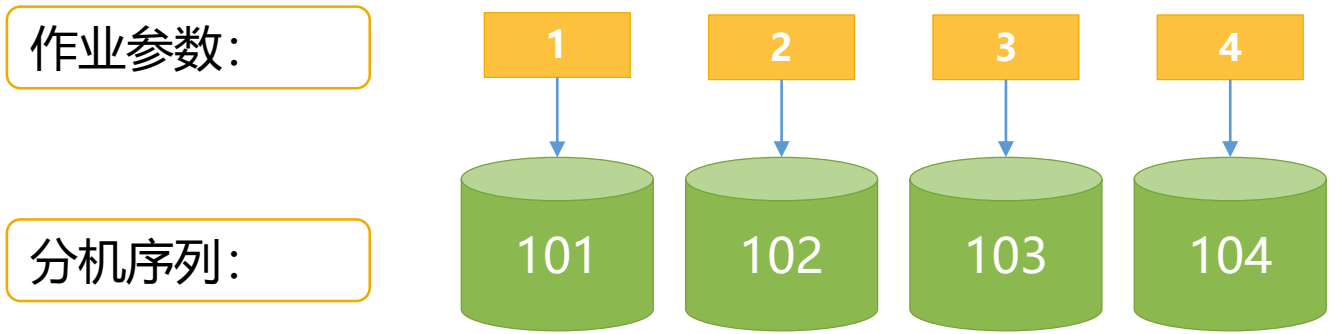
- **等量作业** 当n=m时，我们称之为等量作业。

等量作业一般用于数据拆分的分区号，跟部署分机的台数以及序号严格对应时。

此时的分配算法不再考虑计算均衡，而是直接将作业号分配到对应的分机，第k台分机分配第k个参数。

等量作业的分配算法很简单，即按照作业顺序，第一个作业分到第一台分机，第二个作业分给第二台分机，直到最后一个作业。

callx(“sub.dfx”,[1,2,3,4];A1)的分配如下：



任务分配—大量作业

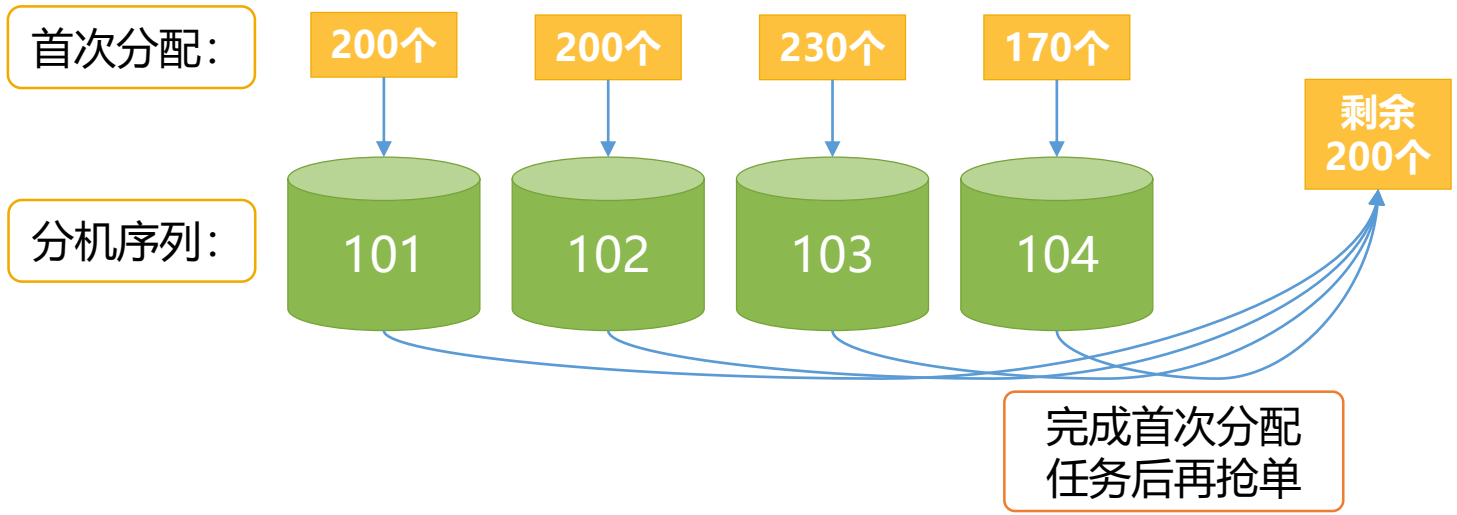
- **大量作业** 当n>m时，我们称之为大量作业。

大数据量计算时，为了更好地控制好并行分机间的负载均衡，往往会将计算拆分为大量小作业。否则，如果作业数太少，碰到某个作业执行时间过长，其它完成作业的分机都只能闲置等待。

而在分配每个作业时，需要先询问每台分机的算力，再决定分给谁。当作业量大到某种程度后，分配作业本身的时间就会过于多了。

为了提高作业的分配效率，callx在询问分机的算力后，根据预估算力，一次性将作业的80%量分配下去。留下20%的作业用于分机的负载均衡，由完成已分配作业的分机来申请，这样可以保证资源的均衡使用。

例如callx(“sub.dfx”,to[1000];A1)的分配：



任务分配—reduce

callx函数的作业计算完成后，会得到所有作业的结果集合，而下一步的工作常常是再做进一步的聚合。如果将这些聚合运算分摊到分机执行，则能得到这样的好处：

1、可以减轻主控机的计算压力，让最后一步聚合运算也可以利用集群来并行处理。
2、分机聚合运算后，多个作业的结果最终被聚合为一个返回结果，跟主控机的网络通讯量显著减少，减轻网络传输压力。

在分机每完成一次作业后将执行一次reduce动作，将作业结果与当前分机聚合结果再运算，分机完成所有作业后，将本机聚合结果发送给主机。

本例虽然有10000个作业，但是使用了reduce表达式后，每台分机会将所有作业的结果聚合为一个结果。

最终，主控机只会得到和处理4个分机结果。

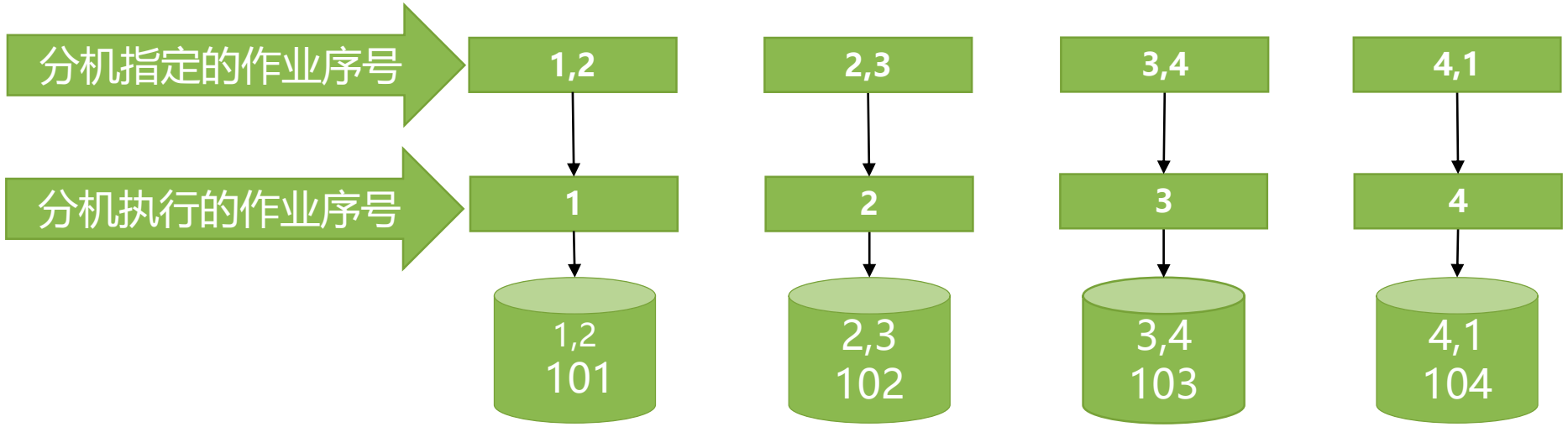
reduce运算可以用一个当前聚合值（用~~表示）跟当前作业结果值（用~表示）的表达式来描述，这个表达式称为**reduce表达式**。下例使用了‘~~+~’ 表达式，将分机中的作业返回结果全部加起来，类似于sum操作：

	A
1	=["192.168.0.101:8281","192.168.0.102:8281","192.168.0.103:8281","192.168.0.104:8281"]
2	=callx(“sub.dfx",to(10000);A1;~~+~)
3	=A2.sum()
4	

任务分配—指定作业

前面讨论中是假定每个分机上都有全量数据，因而可以随意分配作业。但实际场景时分机上常常只有一部分数据，只能执行部分作业，这时可用callx的s参数，来指定作业可分配的分机。

s参数是个数列的序列，写明每个分机可以执行的作业序号。比如4台分机上，按季度分布的数据且容错度为1时（有多一倍的冗余数据），那么此时的s就是当前所有分区的集合，s= [[1,2],[2,3],[3,4],[4,1]]，则callx(“sub.dfx”,to[4];A1,s)的可能计算分配为：



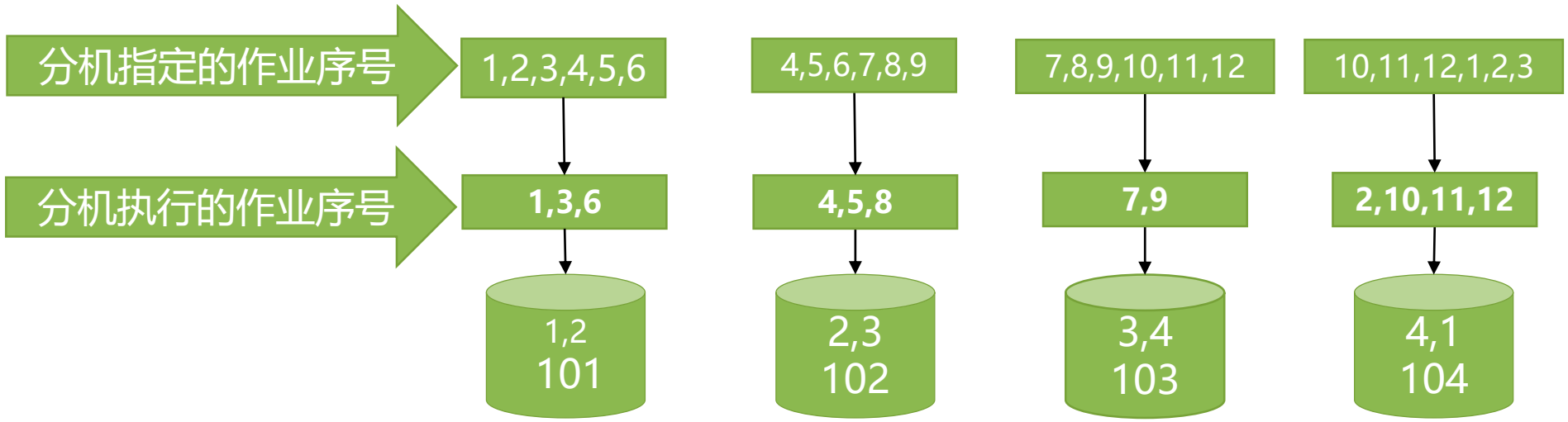
上例中，s通常不需要直接写出，而是可以由zone(A1)直接算出来。

任务分配—指定作业

上页例子中，每一个分区只对应一个作业。而考虑到更好的负载均衡，作业往往需要拆得更细。

那现在用月份来计算时，则每个季度需要对应3个参数。

此时的s参数为 s= [[1,2,3,4,5,6],[4,5,6,7,8,9],[7,8,9,10,11,12],[10,11,12,1,2,3]]，如下为一种可能的计算分配：



同样地，上面的s也不需要直接写出，否则太麻烦了。此时的s可以用z:m简写形式，其中z=zone(A1),m=3时就是上面的s。另外s还有直接m形式，关于这两种形式的详细用法，读者可以参考润乾官方文档。

THANKS

感谢观看