|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 分类号 |  | |  | | | 密 级 | |  |
| U D C |  | |  | | | 编 号 | | XXX |
|  | | | | | | | | |
| C:\Users\liutianjiao\Desktop\TIM图片20191230192855.pngTIM图片20191230192855  **硕 士 学 位 论 文** | | | | | | | | |
|  | | | | | | | | |
| 请输入论文标题 | | | | | | | | |
|  | | 研 究 生 姓 名： | | ： |  | |  | |
| 学 号： | | ： |  | |
| 指导教师姓名及职称： | | ： |  | |
| 专 业 名 称： | | ： |  | |
| 研 究 方 向： | | ： |  | |
| 二〇XX年 月 | | | | | | | | |

# CloudSim云任务调度实验

目录

[CloudSim云任务调度实验 2](#_Toc1406)

[一、 CloudSim介绍 3](#_Toc25243)

[1. CloudSim体系结构](#_Toc8707)[[1]](#_Toc8707) [3](#_Toc8707)

[2. 云计算任务调度概述](#_Toc31107)[[3]](#_Toc31107) [3](#_Toc31107)

[3. CloudSim工作方式](#_Toc9230)[[2]](#_Toc9230) [4](#_Toc9230)

[4. 云服务层共享](#_Toc31418)[[4]](#_Toc31418) [5](#_Toc31418)

[5. 常见的云计算任务调度策略 6](#_Toc30590)

[二、 实现算法介绍 7](#_Toc13257)

[1. 任务调度数学描述](#_Toc2132)[[8]](#_Toc2132) [7](#_Toc2132)

[2. 循环调度算法 7](#_Toc19167)

[3. 贪心调度算法](#_Toc28011)[[5]](#_Toc28011) [7](#_Toc28011)

[4. 遗传调度算法 8](#_Toc1835)

[三、 CloudSim仿真参数配置 10](#_Toc13812)

[1. 虚拟机配置 10](#_Toc29932)

[2. 任务配置 11](#_Toc22454)

[四、 运行结果 11](#_Toc2866)

[1. 运行结果——运行时间和虚拟机上任务均衡度 11](#_Toc27746)

[2. 绘制表格及图表 20](#_Toc26717)

[五、 总结分析 22](#_Toc7803)

[六、 参考文献 22](#_Toc9182)

## CloudSim介绍

### CloudSim体系结构[1]

CloudSim云平台是由澳大利亚墨尔本大学的网格实验室和Gridbus项目宣布推出的云计算仿真软件，是一种开源模拟引擎，基于GridSim和离散事件驱动，可以模拟创建多种云计算环境中的实体，包括云数据中心、物理主机与虚拟机、各组件间的消息传输及时钟管理等。并且，CloudSim作为通用的可扩展的模拟框架，支持模拟新兴的云计算基础设施和管理服务。图 1是CloudSim体系结构。



图 1 CloudSim体系结构

### 云计算任务调度概述[3]

云计算任务的资源调度通常有两种途径：一是物理机将自身计算资源按需分配给相应的计算任务；二是为达到负载均衡的目的，将计算任务迁移到资源利用率低的主机上继续运行。

云计算环境下的任务分配需要综合考虑的因素更多。这些因素主要包括以下几点：

1. QoS要求。满足QoS保障要求是对任务分配策略最基础的要求。任何一种分配策略如果达不到用户任务调度的QoS要求，那么它就无法投入使用，失去实际价值。一般而言，在用户进行任务调度时，会和云服务商签订相应的服务等级协议（SLA）。在该SLA中明确了任务的调度开支、截止时间、可靠性以及安全性等约束条件。实际云计算平台建设中，为给用户提供更加优质的服务从而获得更多的收益，云计算服务商必须充分考虑QoS目标约束要求。
2. 任务完成时间。任务完成时间是衡量任务分配算法好坏的一个十分重要的性能指标。任务完成时间即从用户向云计算中心提交任务到所有任务执行完毕所用时间，在可以接受情况下，完成时间越短，则延时也就越短，用户体验就越好，证明算法执行效率越高，系统性能越好。
3. 负载均衡水平。云计算作为一种商业计算模式，在满足用户任务调度所要求的QoS约束条件下，应充分考虑系统的负载均衡情况，以降低能耗，提高资源使用效率，以最低的成本获得最大的收益。
4. 服务收益。由于云计算规模十分庞大，其用于海量的数据存储和超强的计算能力，在大数据时代，在制定任务分配策略时应该充分分析用户行为，建立相应的数学和经济学模型，更加智能和有针对性的进行任务分配，以获得更高经济利润。

### CloudSim工作方式[2]

CloudSim的工作方式如图 2所示，不同的用户提交的任务是相互独立的，数据中也可以有多个，代表来自不同云供应商的云资源，数据中也包含性能差异的物理机，通过虚拟化技术将物理资源抽象重组成不同性能的虚拟机。CloudSim通过VmAllocationPolicy方法实现虚拟机和主机之间的映射策略。虚拟机之间的共享策略由VmScheduler来实现，包括时间共享策略和空间共享策略。用户自定义的任务和资源的映射方式需要在DatacenterBroker中扩展实现。云信息服务中心（CIS）就是模拟调度的核心。

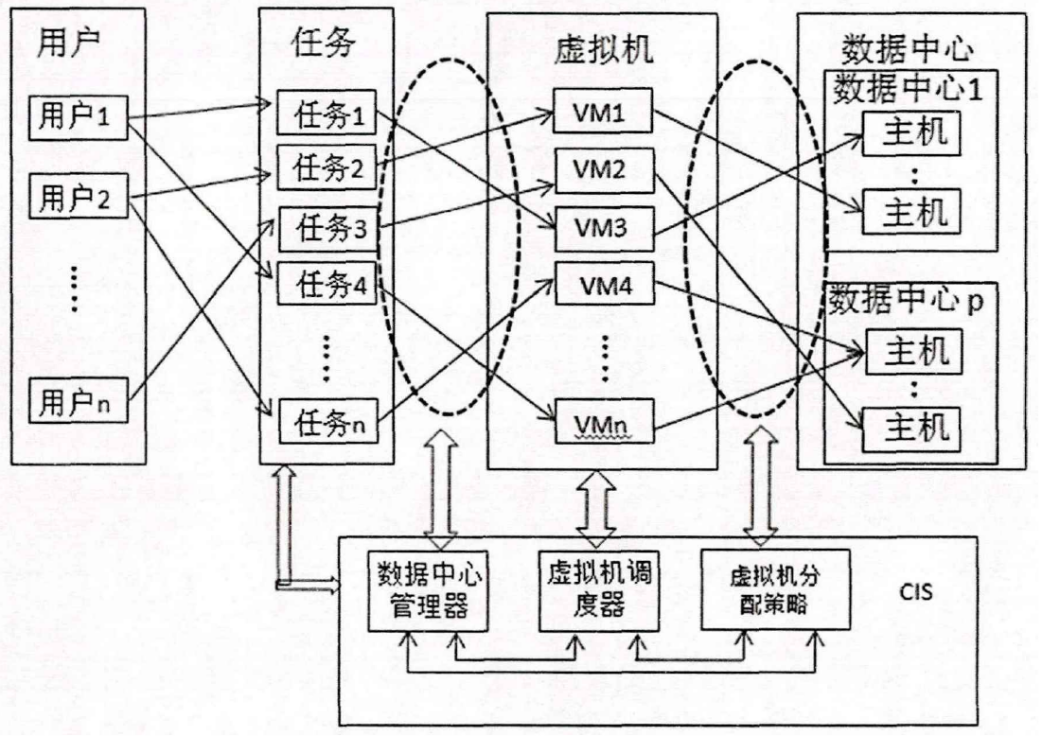


图 2 CloudSim工作方式

由图 2可以看出，在云计算环境下，从用户提交任务请求到获得相应的执行资源过程可以分为两个层次。

第一个层次是任务分配阶段，即用户提交的海量应用任务经过云数据中心代理均衡分配到集群中各虚拟机上执行的过程。该过程实现的是任务到虚拟机的映射，完成任务所需资源和虚拟机提供的虚拟资源之间的匹配。

第二个层次是虚拟机部署迁移阶段，即通过虚拟机合理部署和动态迁移实现对物理资源的高效利用。该阶段实现的是虚拟机到物理宿主机的映射，完成物理资源如何均衡的被虚拟机利用。

通过对CloudSim的工作方式和自带案例分析得知，任务调度在云平台仿真系统中的一般流程如下：

1. 初始化CloudSim的环境，初始化用户数量、日历和标志；

CloudSim.init(num\_user, calendar, trace\_flag);

1. 创建数据中心。数据中心由多个主机组成，一个主机代表一个或多个虚拟机，数据中心还可以设置不同的虚拟机调度策略；

Datacenter datacenter0 = createDatacenter("Datacenter\_0");

1. 创建数据中心代理；

创建数据中心代理。数据中心代理会管理用户提交的任务和数据中心的虚拟机，利用调度策略完成调度。本文编写的算法主要在此处进行调用，完成调度。

DatacenterBroker broker = createBroker();

1. 创建虚拟机列表。对虚拟机参数进行设置；

Vmlist = createVM(brokerId,)

1. 创建云任务列表；

Cloudlet cloudlet = new Cloudlet(id, lengths[i], pes Number, filesize, output Size, utilization Model, utilization Model, utilization Model);

1. 根据自定义的调度策略，讲云任务绑定到虚拟机；

Broker.bindCloudletToVM(Cloudletlist, Vmlist);

1. 开始仿真；

CloudSim.startSimulation();

1. 结束仿真。

CloudSim.stopSimulation();

云计算的任务调度，实质上是任务对云计算中的资源节点进行竞争，因此，云计算没办法对每一个任务都提供最好的资源，所以该类型的任务调度是NPC问题，即在任务数量增加时，求解最佳解决方案所需要的时间呈指数级增长。

### 云服务层共享[4]

尽管VM上下文是相互独立的（通常指主存和辅存空间），但仍会共享CPU内核和系统总线。因此，VM的可用资源仍受主机处理能力限制。为了实现不同环境下对不同调度策略的模拟，目前的CloudSim支持两层VM调度：主机层和VM层。主机层中的VM调度直接指定VM可获取的处理能力，而VM层中，VM为在其内运行的独立任务单元分配固定的处理能力。

两层VM调度均实现了时间共享和空间共享。以下分析两者在应用任务调度性能上的区别。如图2，某主机可运行两个VM，该主机拥有两个CPU内核，每个VM请求两个内核并执行四个任务，、占用VM1，而占用VM2。

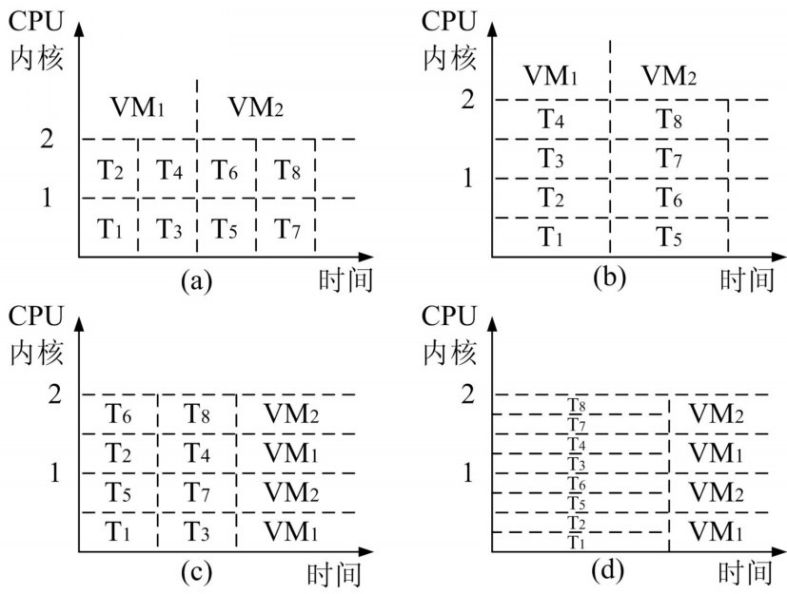


图 3 四种VM调度场景

图 3(a)中，VM主机层和任务单元均采用空间共享。空间共享中，VM请求两个CPU内核，对应时间内只能运行一个VM。同样的原因，对于VM1，其任务单元分配模式也是一样的，由于一个任务单元只需一个CPU内核，因此可以同时运行任务单元 T1 和T2。此时，T3和T4 队列中等候。

图 3(b)中VM采用空间共享，任务单元采用时间共享。

图 3(c)中VM采用时间共享，任务单元采用空间共享。此时，内核通过时间片原理将其处理能力在VM中进行分配，而时间片本身以空间共享方式分配至任务。由于内核共享，VM 的可用处理能力是变化的。而任务单元是空间共享，表明内核只执行一个任务单元。

图 3(d)中VM和任务单元均采用时间共享。此时，VM共享处理能力，并且同时将共享内核分配至所有任务单元。

### 常见的云计算任务调度策略

云计算相关实现技术及软件多采用比较成熟和简单的任务分配策略，以Hadoop为例，其采用最多的还是经典的FIFO算法，此外，还有Facebook公司为满足并行执行多种任务的需要而提出的公平份额调度算法和Yahooo公司提出更够有效管理集群资源的计算能力调度算法等。

目前，科研人员基于CloudSim实现了多种方式的任务分配策略。文献[5]综合考虑云任务与虚拟机资源的特征，提出了一种改进的贪心策略。文献[6]通过使用优化后的蚁群算法实现了任务调度，实现了降低任务执行时间的同时提高虚拟机资源的负载。文献[7]把任务执行时间和系统负载均衡度作为优化目标，提出了一种基于混沌猫群算法实现多目标调度。文献[2]基于粒子群算法实现了任务调度。文献[8]将遗传算法和模拟退火算法结合，去各自优点，实现了任务任务调度。文献[9]将蚁群算法和模拟退火算法相结合实现了任务调度。文献[1]基于蚁群算法，综合考虑SLA服务保证和负载均衡实现任务分配，引入了基于灰色预测模型的虚拟机迁移算法。文献[3]将遗传算法和蚁群算法结合，实现了任务调度。

## 实现算法介绍

### 1. 任务调度数学描述[8]

虚拟资源节点由表示，其中m表示虚拟资源节点的数量，表示第j个可供选择的资源节点。

切割后的子任务用表示，其中n表示子任务的数量，表示第i个需要执行的子任务。

表示任务i在虚拟资源节点j上执行完毕所需要的时间。表示计算任务i所需要的指令数，表示虚拟资源节点j的平均执行速度，那么可以表示为：

表示任务i传输到虚拟资源节点j所需时间。

表示任务i的数据量大小，表示虚拟资源节点j的带宽，那么可以表示为：

表示由虚拟资源节点完成任务i所需要的全部时间，其计算方式可以表示为：

由于云环境中虚拟资源节点是可以并行工作的，所以每个资源节点独立完成自己的工作，系统处理完所有子任务所需要的时间为：

### 循环调度算法

CloudSim默认实现了循环调度策略，循环调度策略是把一组任务顺序分配给一组虚拟机，尽量保证每个虚拟机运行相同数量的任务以平衡负载，但没有考虑任务的需求和虚拟机之间的差别。

### 贪心调度算法[5]

Min-Min云计算任务调度算法采用先易后难的策略，先执行完成时间短的任务，然后执行完成时间长的任务，并采取贪心策略把每个任务优先指派给执行它最早完成的计算资源。

Max-Min云计算任务调度算法则恰恰相反，采用先难后易和贪心策略，每次选取完成时间最长的任务，再执行完成时间短的任务，并采取贪心策略把每个任务优先指派给执行它最早完成的计算资源。

### 遗传调度算法

#### 4.1 遗传算法[3]执行流程见图 1。

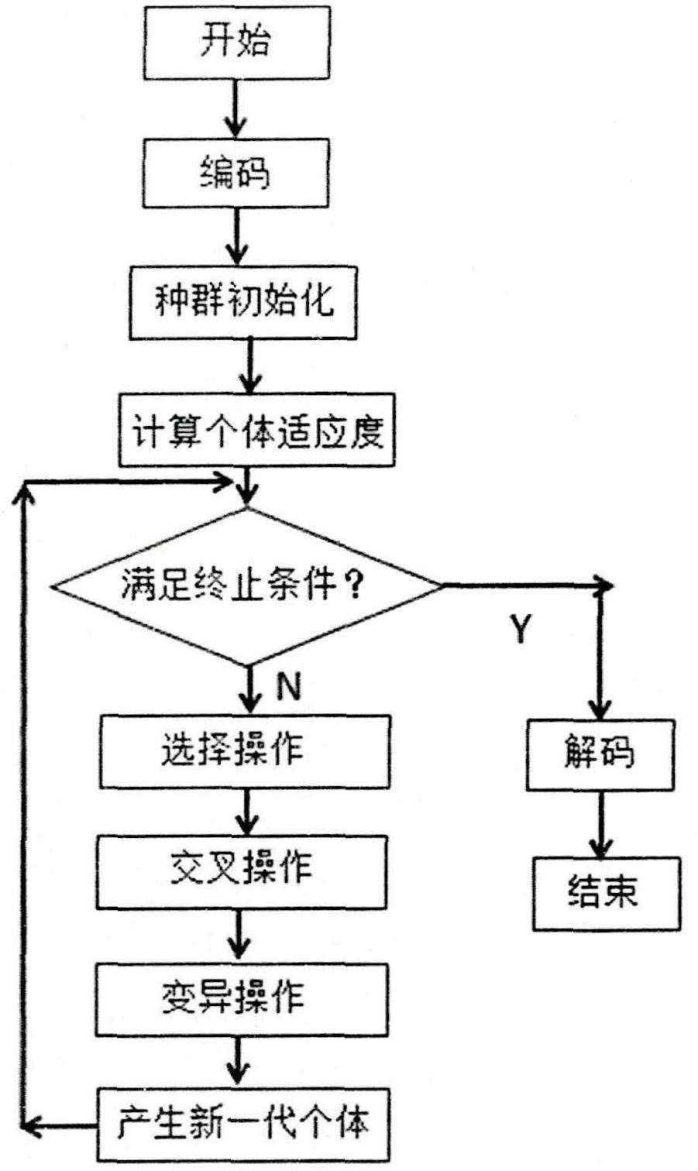


图 4 遗传算法执行流程图

#### 4.2 遗传算法求解任务调度的数学描述[3]

1. 遗传编码

使用遗传算法求解问题的第一步是对所求问题进行编码，即将实际问题中的参数转换成染色体的形式。介于二进制编码存在长度过长，求解过程占用内存过多的缺点，本文采用实数编码的方式进行编码。根据前文中云计算中资源分配模型的设定，本文研究问题假设有ｎ个子任务和ｍ个虚拟资源节点，那么染色体的总长度即为子任务的个数ｎ，染色体的每一位取值范围为[1,m]。假设n=20，m=5，则染色体总长度为20，每一位取值为1~5之间的随机数，可假定一组编码实例为{2，4，2，3，1，3，5，5，3，2，4，1，4，3，1，5，3，2，5，5}对上述染色体解码为：

资源节点1：｛子任务5，子任务12，子任务15｝

资源节点2：｛子任务1，子任务3，子任务10，子任务14，子任务17｝

资源节点3：｛子任务2，子任务11，子任务13｝

资源节点4：｛子任务7，子任务8，子任务16，子任务19，子任务20｝

从上面的解码可以得到每个虚拟资源所执行的任务情况，通过获取每个子任务的完成时间就可以通过前面公式求得所有任务的总完成时间。

1. 初始化种群

解决了遗传算法的编码问题后，需要对所求问题进行种群的初始化。遗传算法中种群规模需要提前设定，它决定了每次迭代后种群中的个体数，初始种群一般是用完全随机的方式生成种群规模２倍的个体，然后通过一定的规则筛选出适应度高的个体组成初始种群。本文中设定种群规模为scale，子任务个数为n，虚拟资源数为m。则种群初始化过程为：随机生成scale×3个个体，然后筛选出scale个适应度高的个体，个体的染色体长度均为ｎ，染色体每一位的取值范围为[1，m]。

1. 适应度函数

初代种群确定之后，算法会通过适应度函数来对所有染色体进行优劣评估。每一个个体都有自己对应的适应度值，由此可以确定个体的优劣性，从而为下一步的选择做准备。在本文中，适应度高的个体代表对任务的调度方案效率越高。适应度函数如果选取不当，会导致前期适应度高的个体被选择的概率增加，影响算法的全局搜索能力；后期还容易导致快速产生近似最优解。同时由于算法对适应度函数调用的频繁性，所以对其设计应尽量简单，一般选用目标函数的倒数作为适应度函数：

遗传算法的核心就是通过模拟自然界中生物进化的方式达到搜索最优解的目的，交叉操作可以提高算法的全局搜索能力，变异操作决定了算法的局部搜索能力，选择操作则可以根据一定的规则，对经过交叉和变异后的新种群进行筛选，保留适应度高的个体组成新的种群，这样算法就可以通过迭代，实现对最优解的寻找。遗传算法对最优解的寻找是有方向的，并不是盲目的搜索，同时交叉和变异概率的取值也和算法的搜索能力密切相关。

1. 选择操作

选择操作可以将子代中适应度高的解保留下来，为后面的交叉和遗传操作提供优秀解。选择操作通过计算每一个个体的适应度值，按照一定的规则将适应度低的个体淘汰，即“优胜劣汰”。保存下来的优值解又称为精英解，它们有进行下一步交叉和变异的权利，它们的子代适应度同样很高。本文主要通过轮盘赌的方式进行个体选择，即适应度高的个体被选择的概率大。

1. 交叉操作

遗传算法全局搜索能力的实现主要靠交叉操作。交叉操作发生在两个染色体上，通过两个染色体之间的基因进行交叉和重组，实现产生新个体的目的。本文采用的是实数编码的方式，基因的取值即代表虚拟资源的序号，所以考虑采用两点交叉的方式进行操作。

假设父代个体为和。两点交叉的位置为i和j，则交叉后子代为：

1. 变异操作

变异操作通过对少量染色体进行改变，可以实现算法的局部搜索并加速最优解收敛。因为染色体的每一位都代表一个虚拟资源，所以进行过变异操作时应尽可能少的对基因进行修改。且进行修改时，基因的取值范围为[1,m]。

假设父代个体为：，对进行变异操作，所得子代为：。

传统的遗传算法常使用固定值作为变异的概率，这样容易使算法陷入局部最优状，本次实验采用固定交叉概率5%。

1. 停止准则

由于交叉和变异的特性，遗传算法会不断产生新的个体，选择适当的循环结束准则可以减少不必要的内存和时间幵销。为算法设置迭代次数较简单的方式，但是也有可能会进行不必要的迭代。本次实验采用固定迭代次数结束迭代。

#### 4.3 遗传算法的优缺点：

遗传算法的优点：

1. 遗传算法具有良好的并行性。因为每个染色体代表一个解，在进行适应度计算时可以同时对染色体进行求解。这样可以节省算法的执行时间。
2. 具有自组织、自适应和学习性。通过给算法设置合适的适应度函数，就可对种群提供正确的优化方向，经过交叉编译后的新种群也会在适应度函数的作用下不断朝着最优的方向进化，避免陷入局部最优。
3. 通过交叉和变异操作，算法可以同时具有良好的全局搜索能力和局部搜索能力。
4. 算法通过概率机制进行迭代，増加了算法的鲁棒性。
5. 算法简单容易理解，且可扩展性高，易于其他算法融合。
6. 应用范围广泛，在流行的机器学习和人工智能领域都有应用。

遗传算法的缺点：

1. 算法的编码和解码操作是不可省略的，编码方式的好坏对求解结果有很大影响。
2. 算法是在初始种群的基础上进行交叉和变异操作的，初始种群的选取会影响求解质量。
3. 算法需要用到的参数较多，目前参数的选取主要靠经验，参数选取的好坏同样会影响求解质量。
4. 算法设计的计算量较大，当问题规模达到一定程度时会增加求解时间。
5. 初期如果出现超级染色体，会使得种群多样性减少，算法过早收敛，形成局部最优解。

## CloudSim仿真参数配置

### 虚拟机配置

在CloudSim平台配置数据中心相关参数，其中虚拟机的关键参数如表 2所示：

表 1 虚拟机参数信息

|  |  |  |
| --- | --- | --- |
| 序号 | MIPS | 带宽 |
| 1 | 278 | 1000 |
| 2 | 289 | 1200 |
| 3 | 132 | 1100 |
| 4 | 209 | 1200 |
| 5 | 286 | 900 |

### 云任务配置

初始40个任务配置方案如表 2所示。随后利用随机函数生成100、150、200、300、400、500个任务，任务指令长度分布在1000~5000，任务大小分布在10000~30000之间，通过设置随机种子来让每次生成的随机数相同。

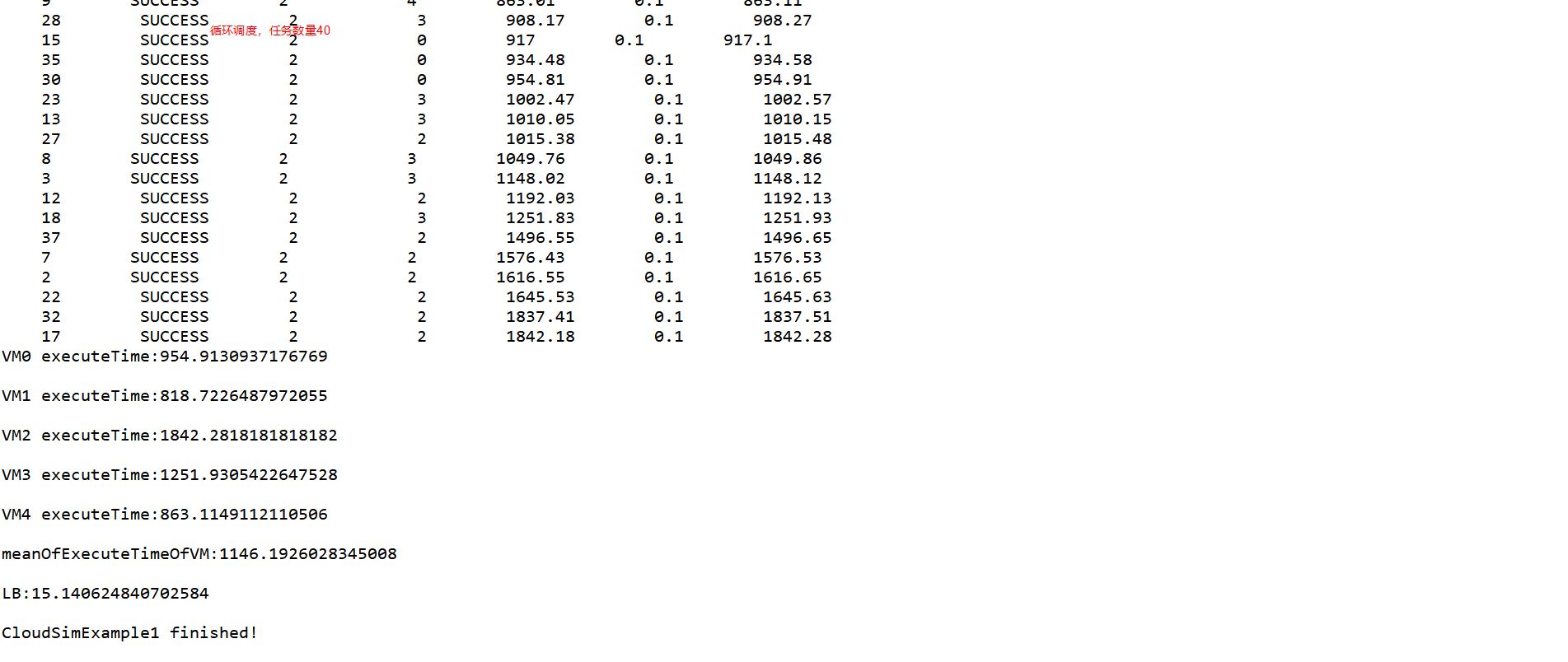
表 2 云任务相关参数

|  |  |  |
| --- | --- | --- |
| 任务数量 | 任务指令长度 | 任务文件长度 |
| 40 | 19365, 49809, 30218, 44157, 16754, 26785, 12348, 28894, 33889, 58967, 35045, 12236, 20085, 31123, 32227, 41727, 51017, 44787, 65854, 39836, 18336, 20047, 31493, 30727, 31017, 30218, 44157, 16754, 26785, 12348, 49809, 30218, 44157, 16754, 26785, 44157, 16754, 26785, 12348, 28894 | 30000, 50000, 10000, 40000, 20000, 41000, 27000, 43000, 36000, 33000, 23000, 22000, 41000, 42000, 24000, 23000, 36000, 42000, 46000, 33000, 23000, 22000, 41000, 42000, 50000, 10000, 40000, 20000, 41000, 10000, 40000, 20000, 41000, 27000, 30000, 50000, 10000, 40000, 20000, 17000 |

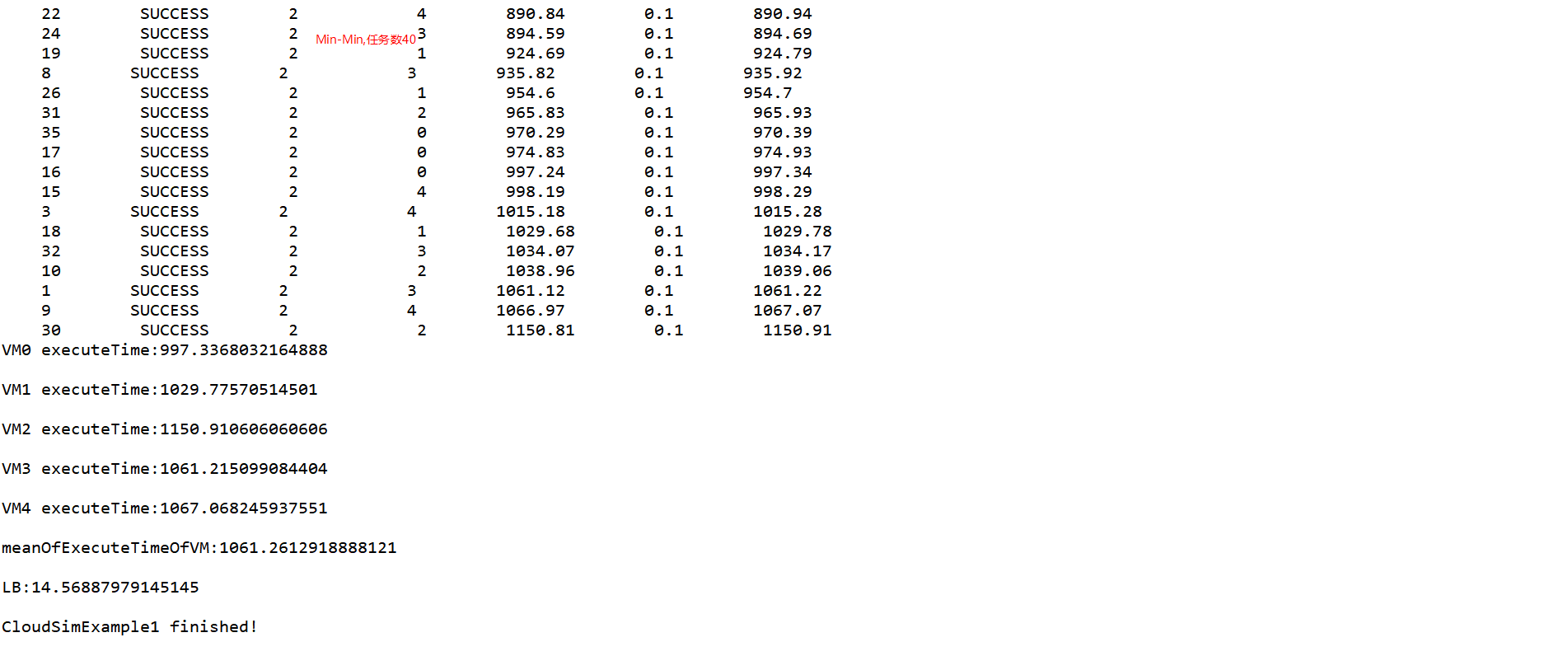
## 运行结果

### 运行结果——运行时间和虚拟机上任务均衡度

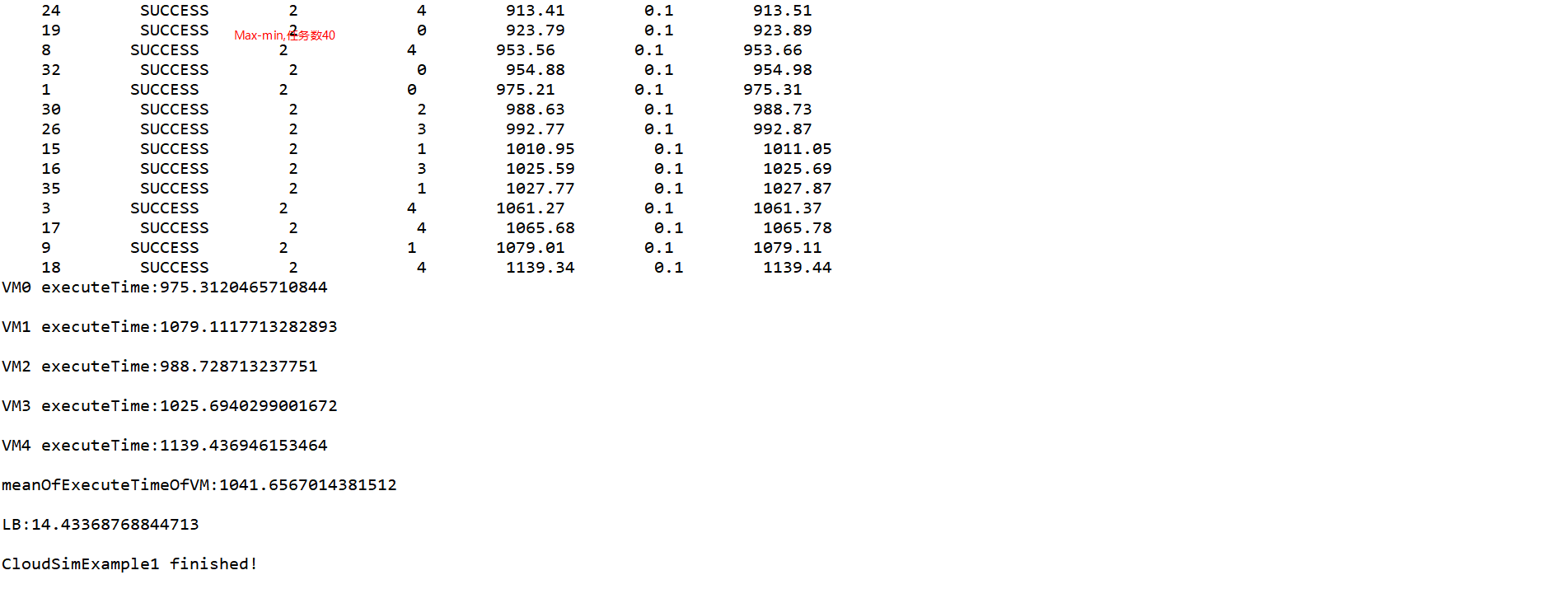
1. 任务数量40个
   1. 循环调度



* 1. Min-min

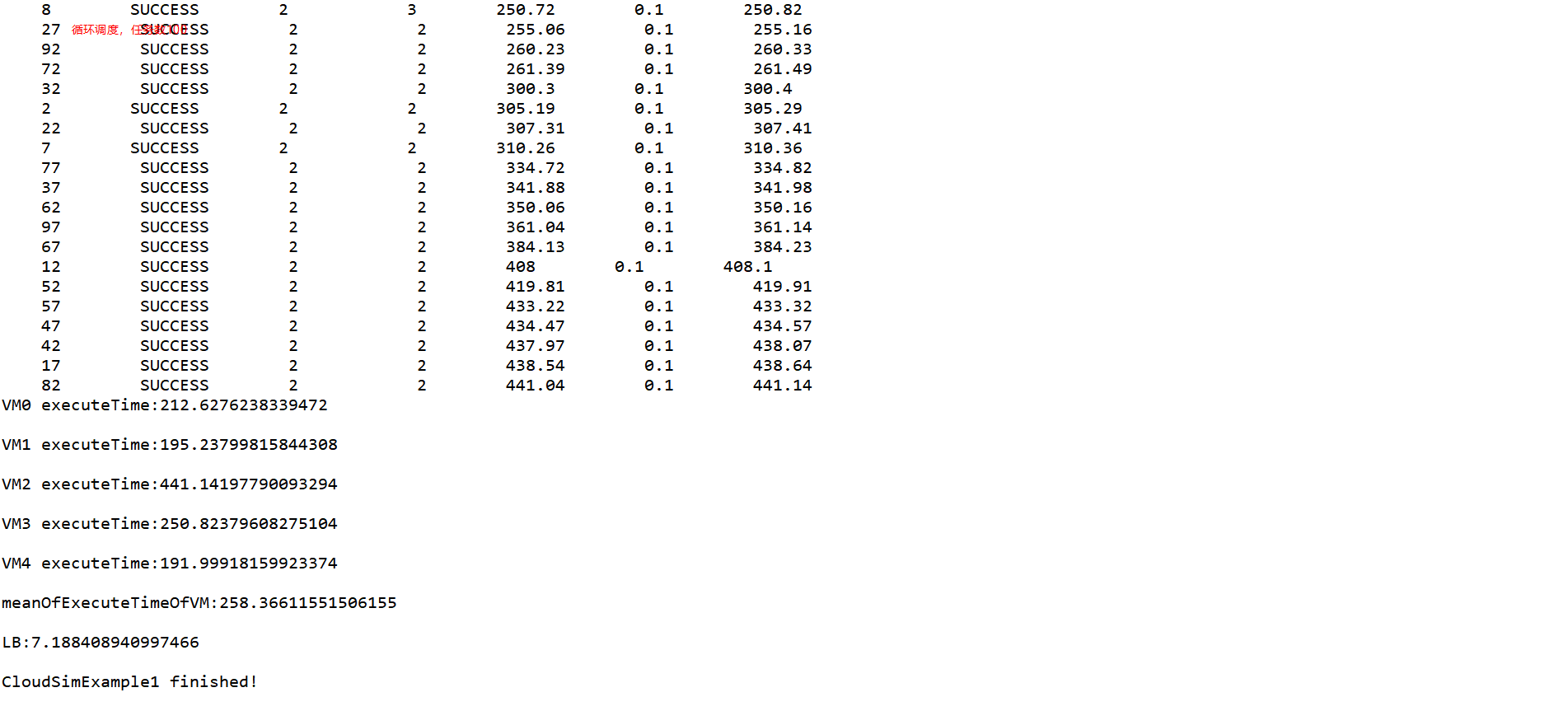


* 1. Max-min

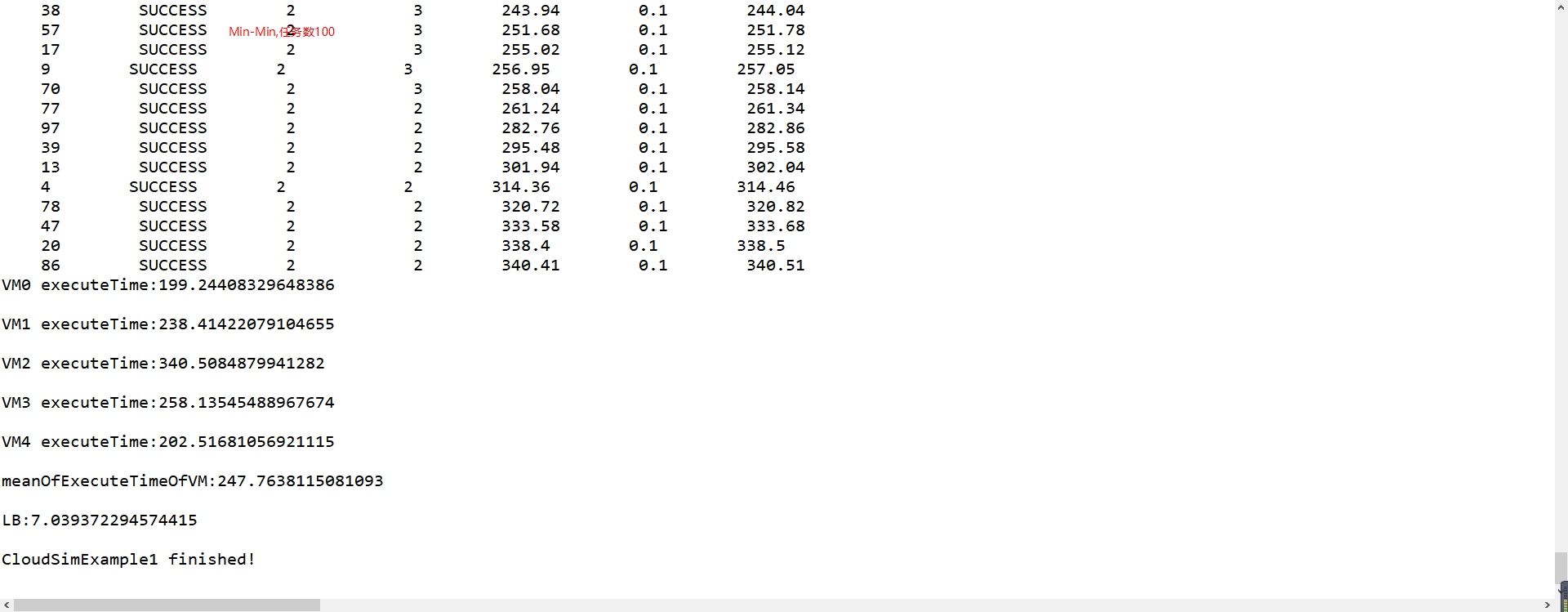


* 1. 遗传算法

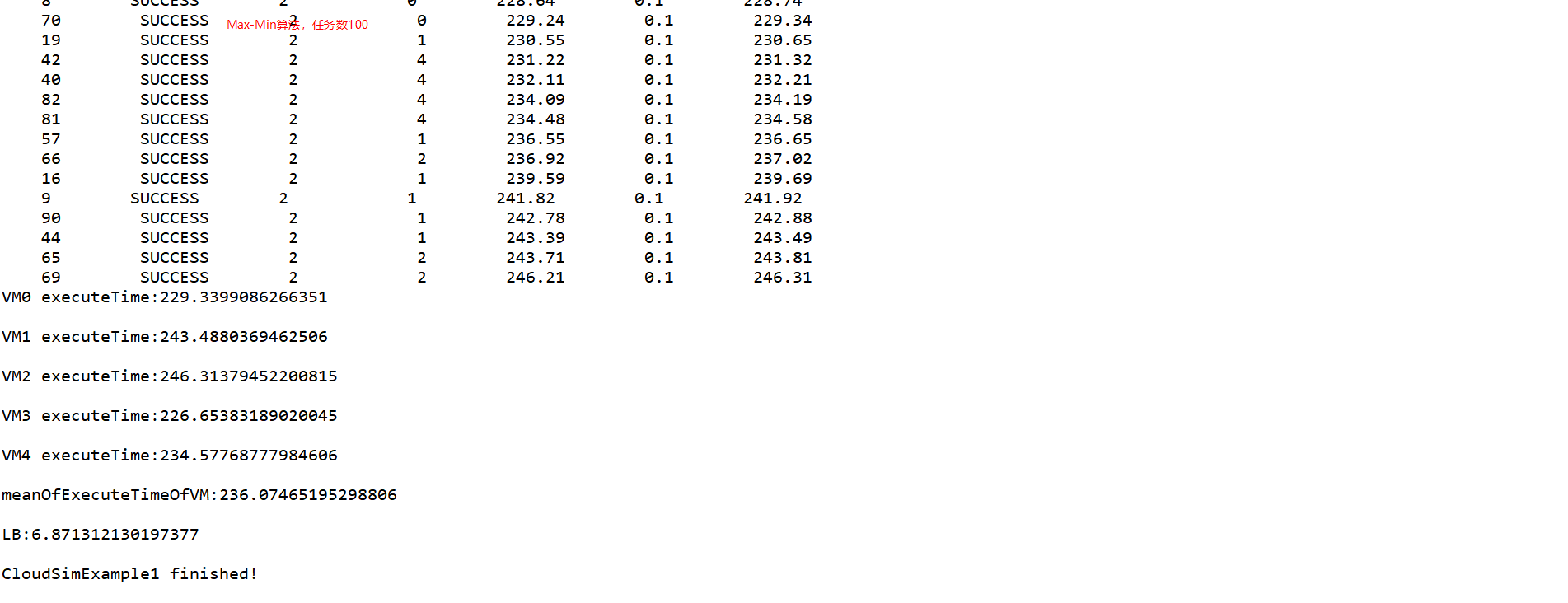
1. 任务数量100个
   1. 循环调度



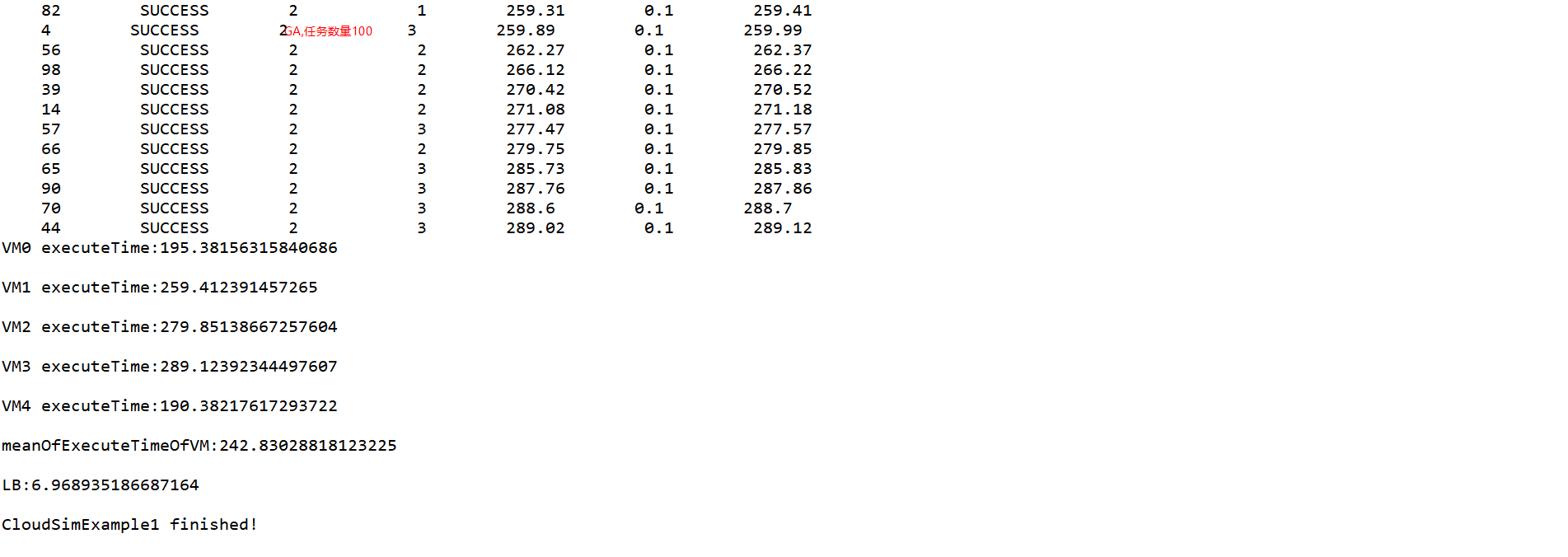
* 1. Min-min



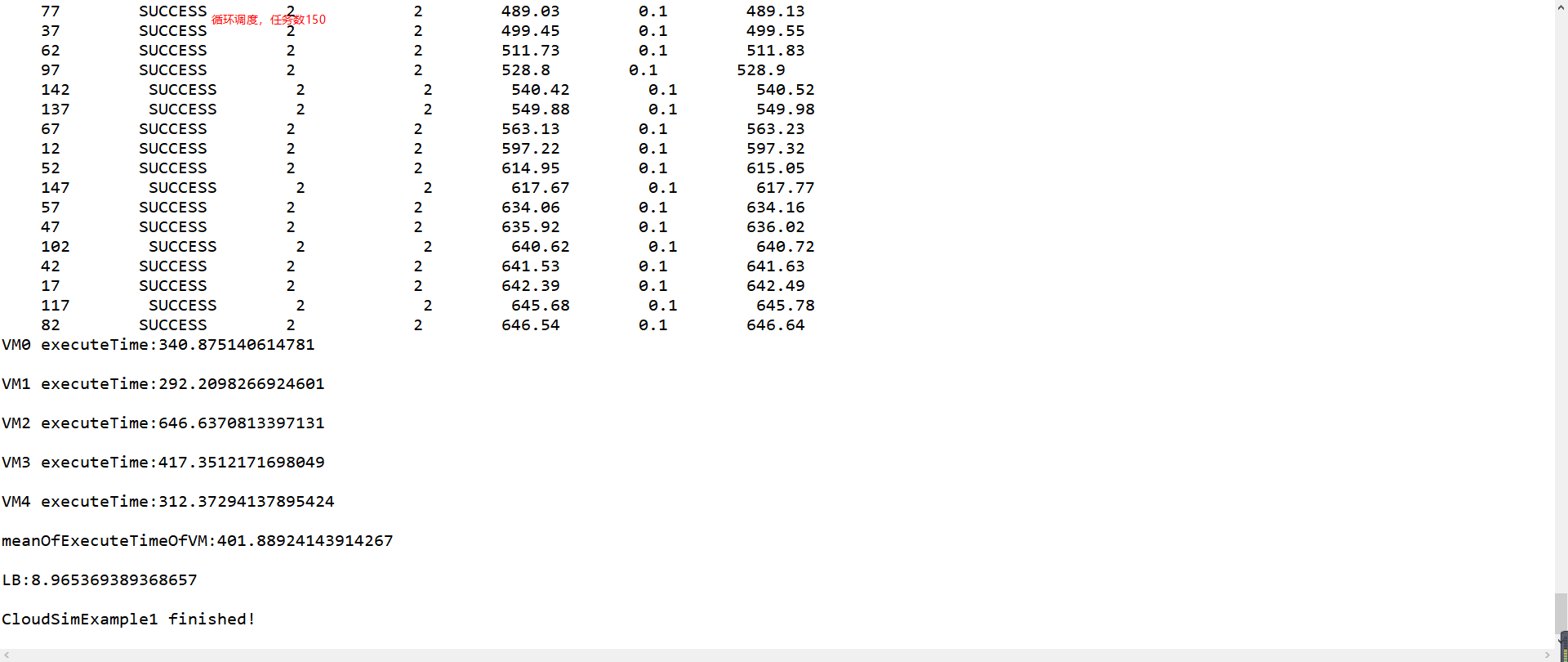
* 1. Max-min



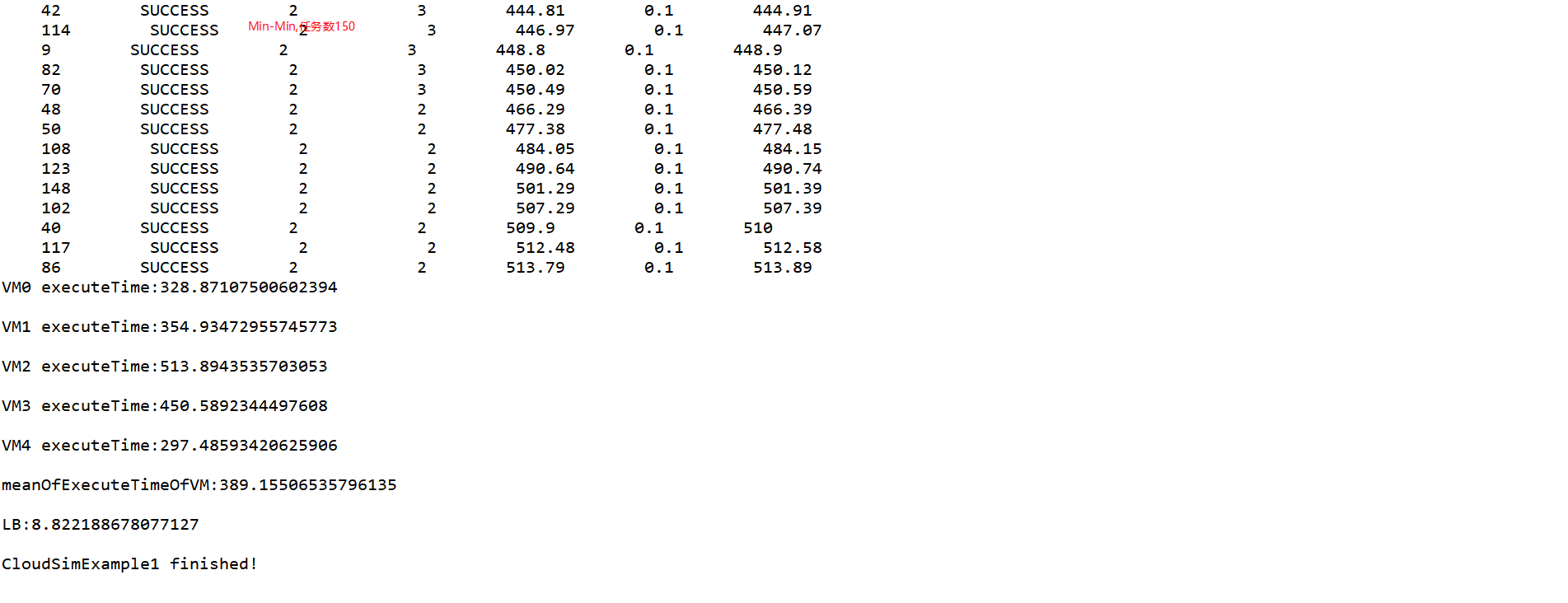
* 1. 遗传算法



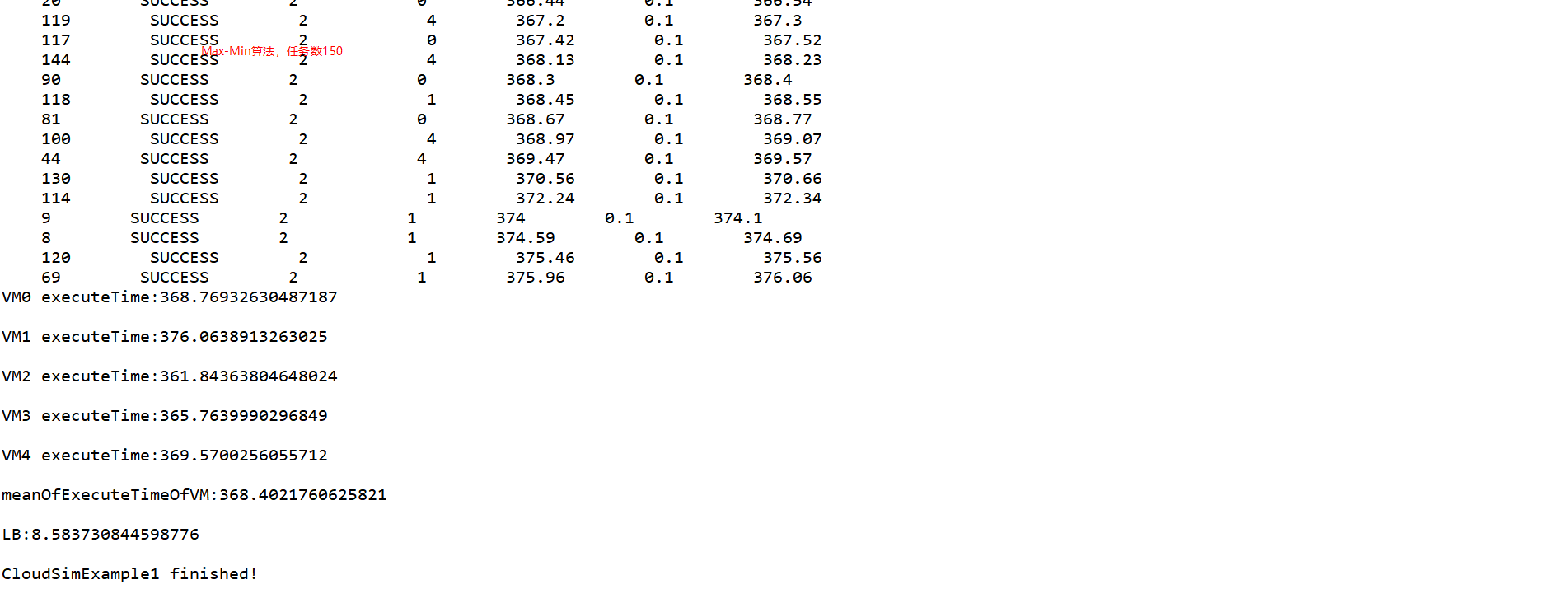
1. 任务数量150个
   1. 循环调度



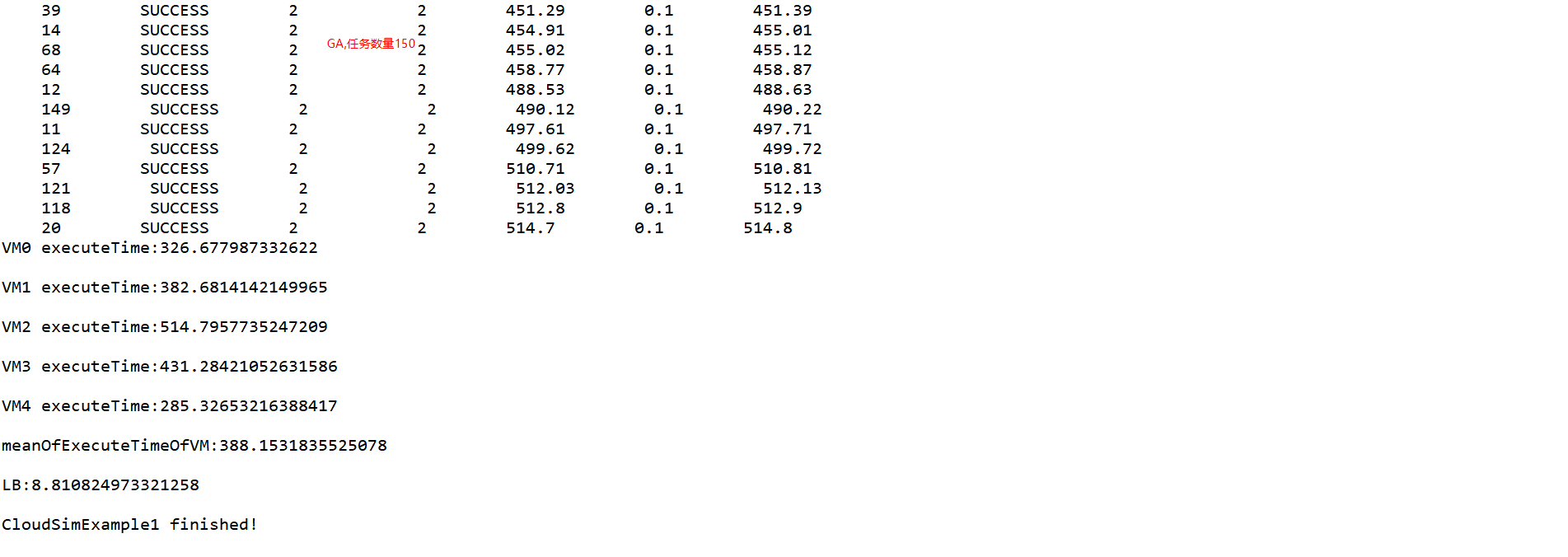
* 1. Min-min



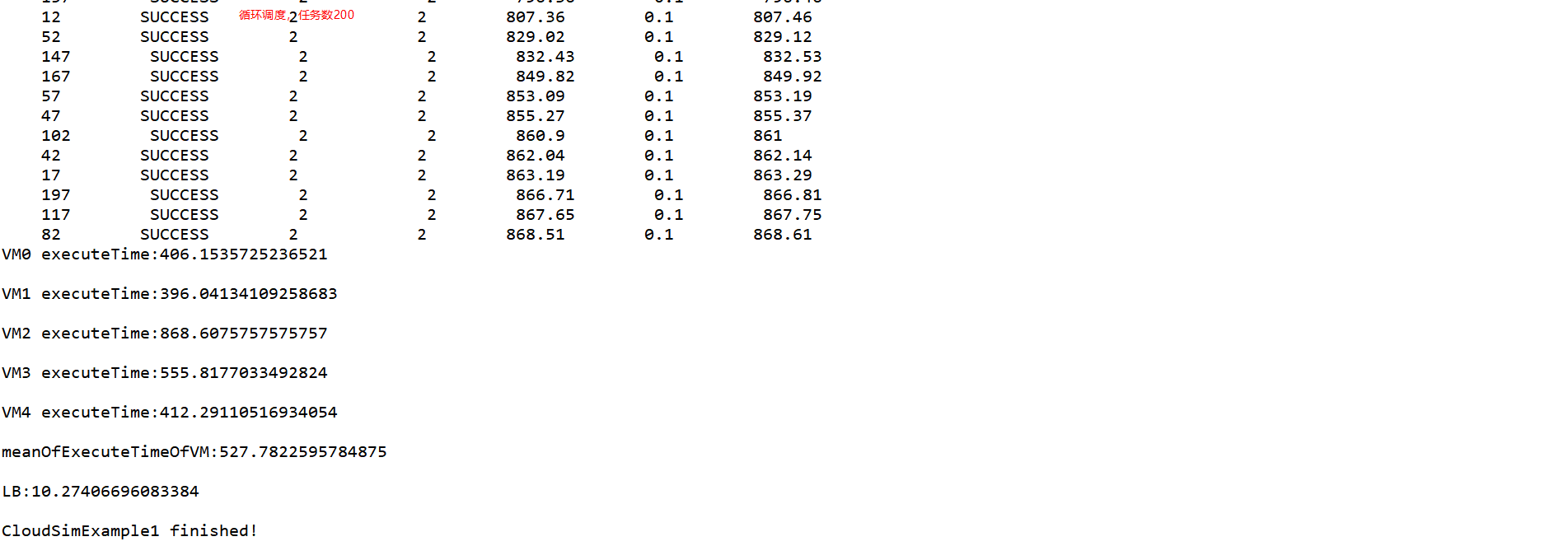
* 1. Max-min



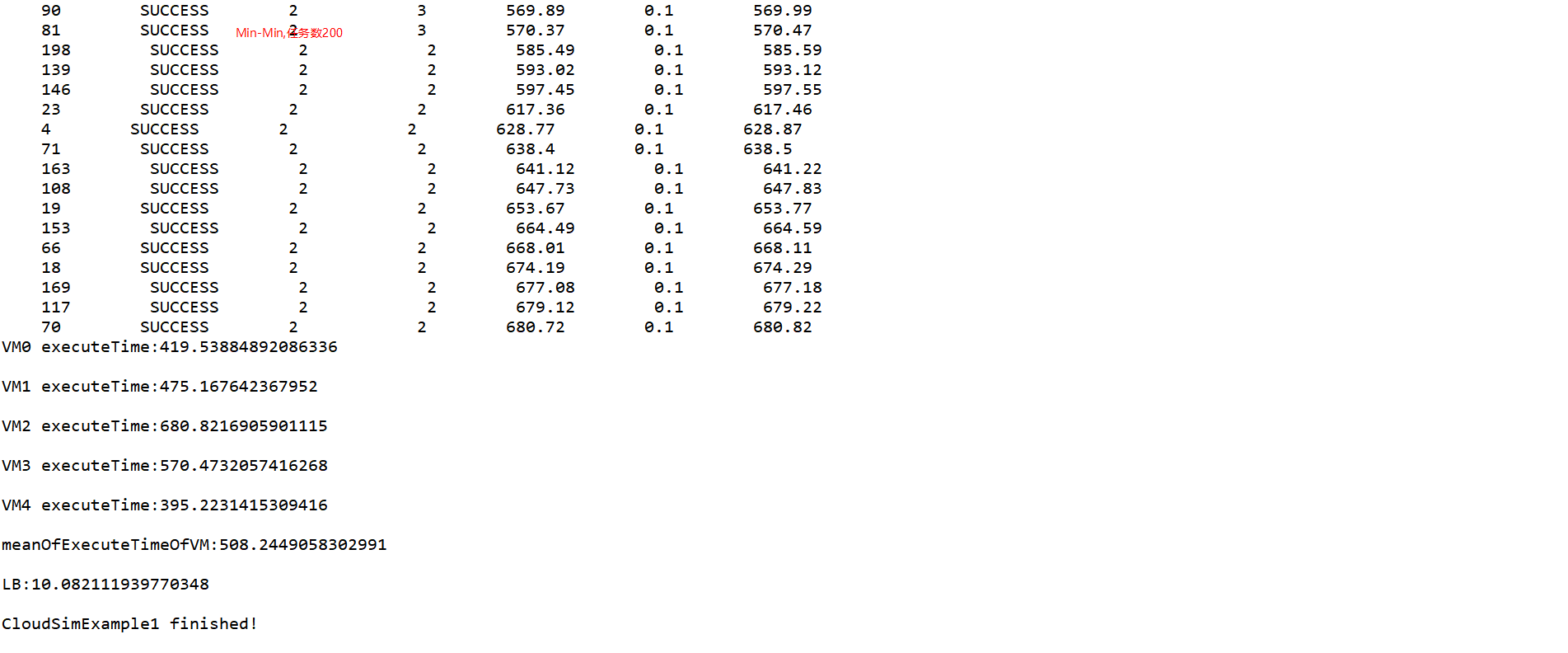
* 1. 遗传算法



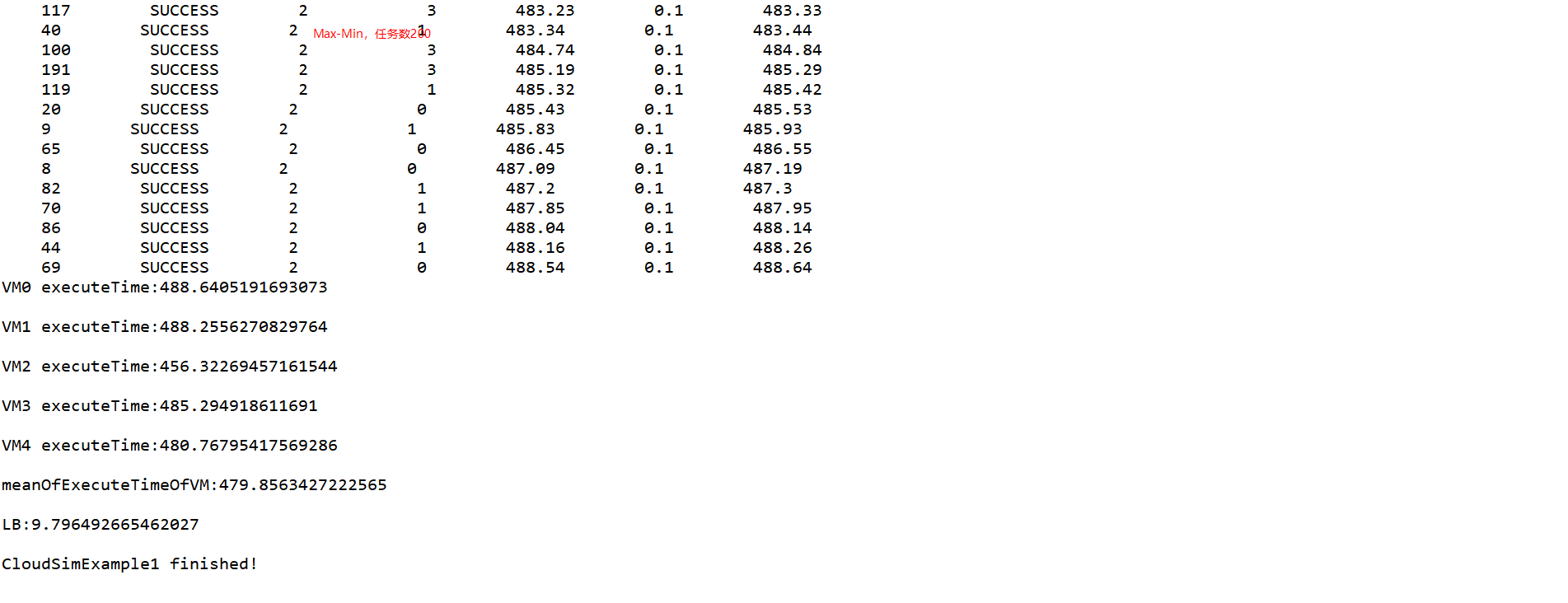
1. 任务数量200个
   1. 循环调度



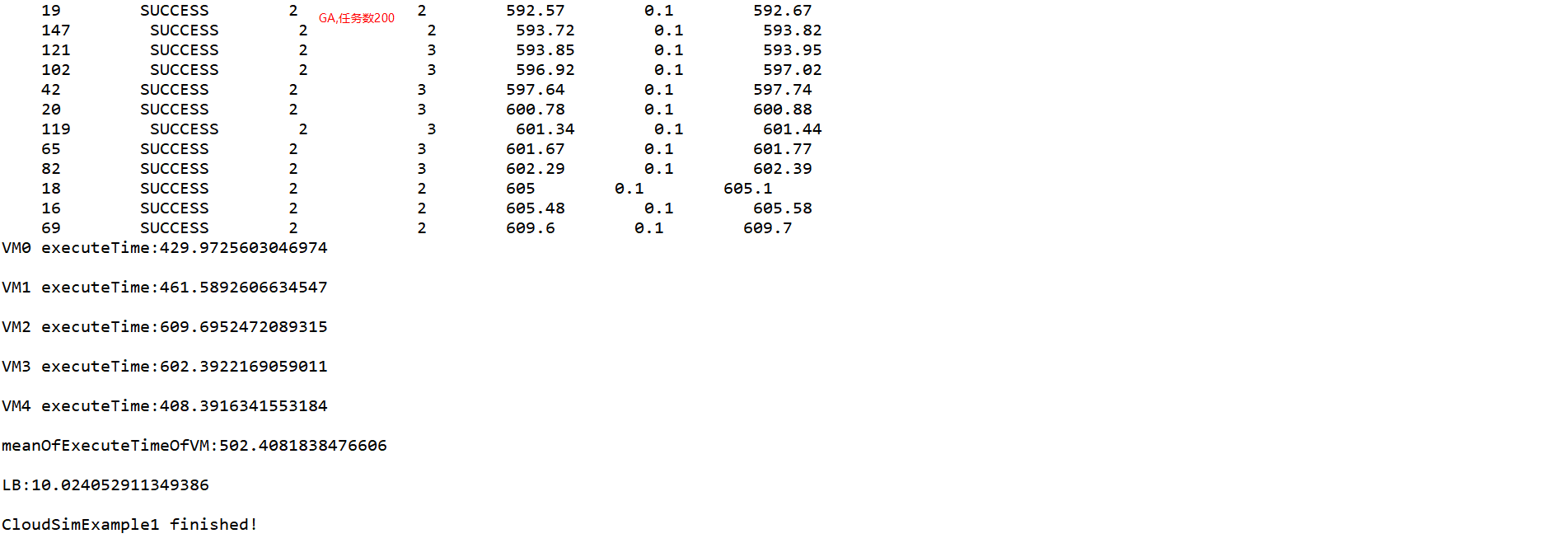
* 1. Min-min



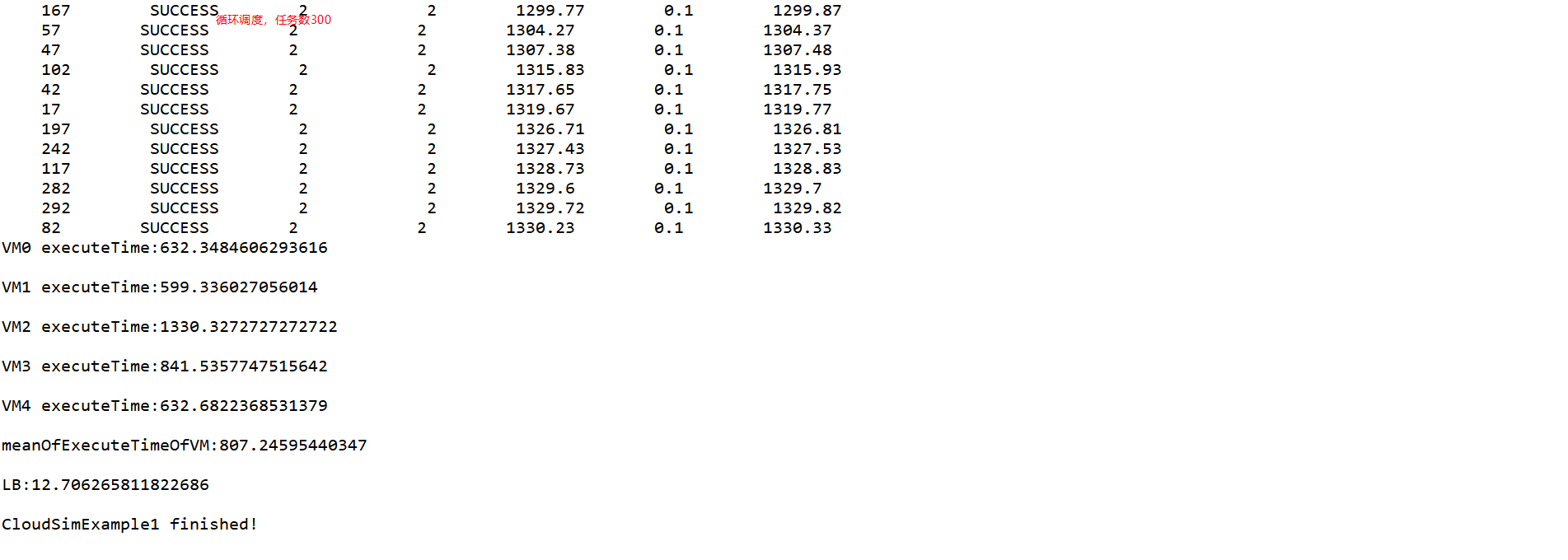
* 1. Max-min



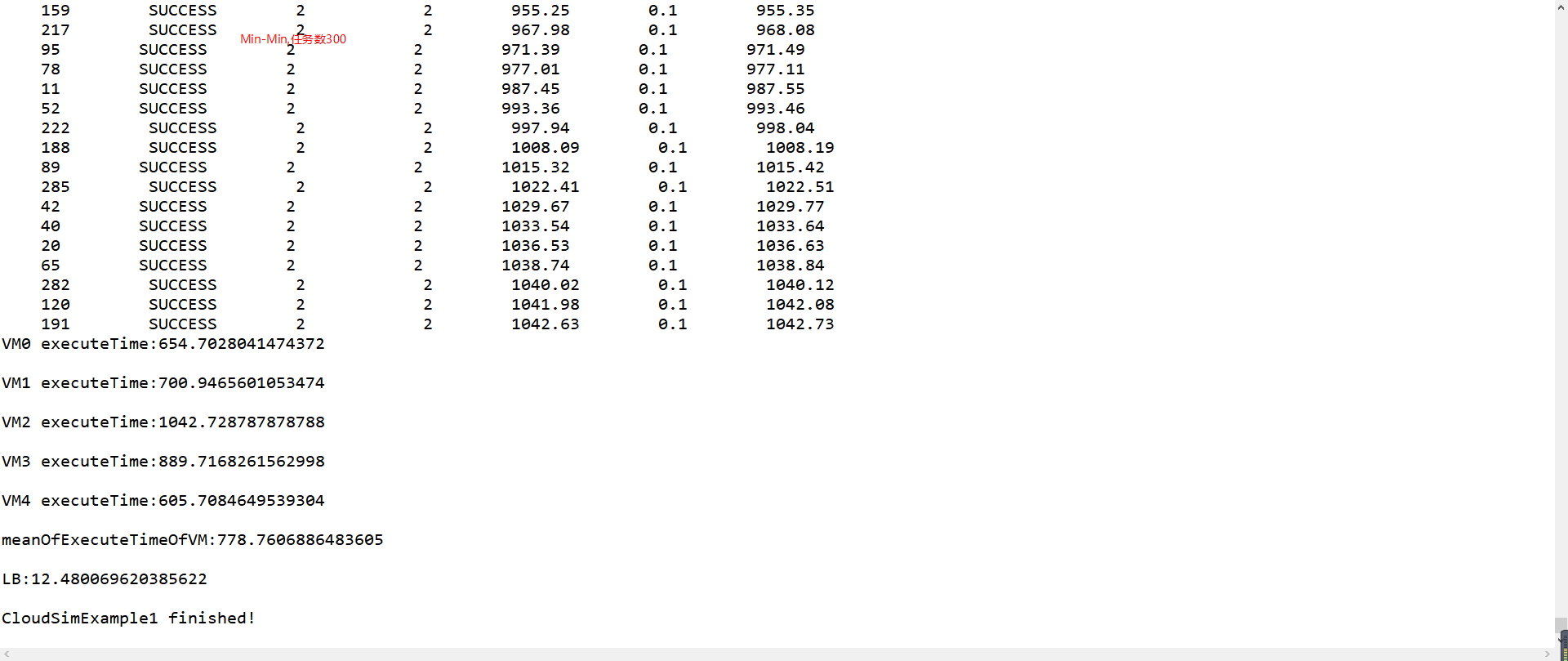
* 1. 遗传算法



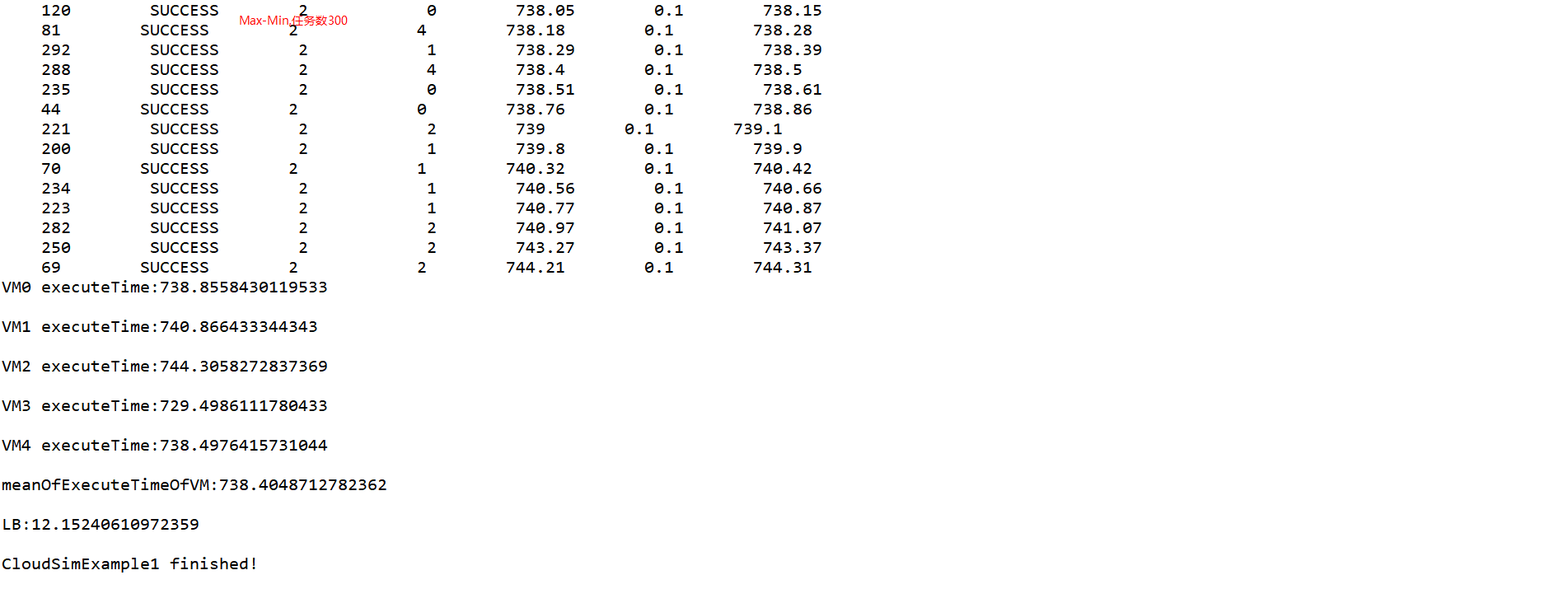
1. 任务数量300个
   1. 循环调度



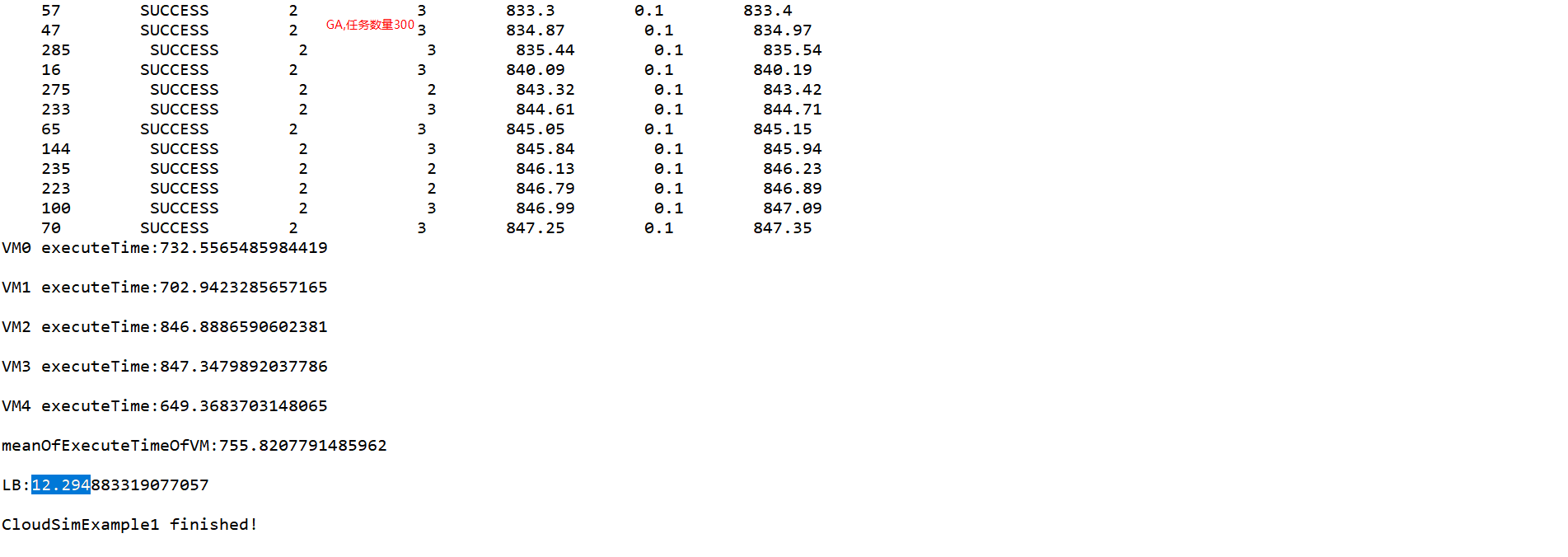
* 1. Min-min



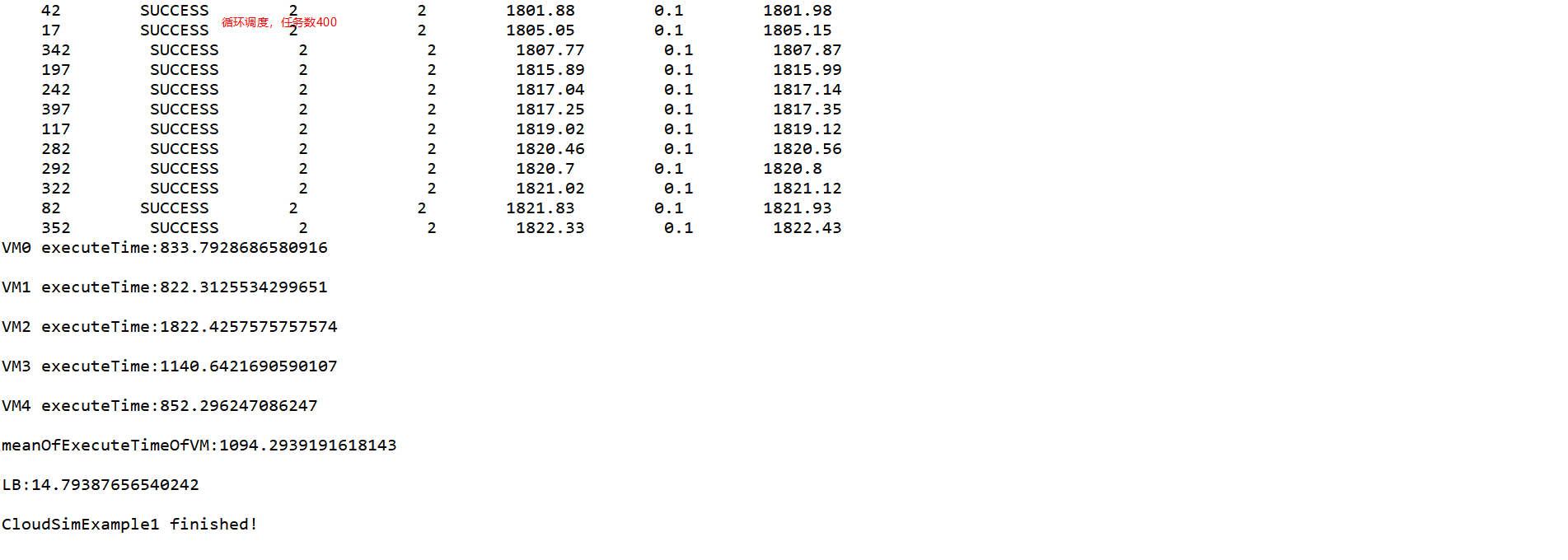
* 1. Max-min



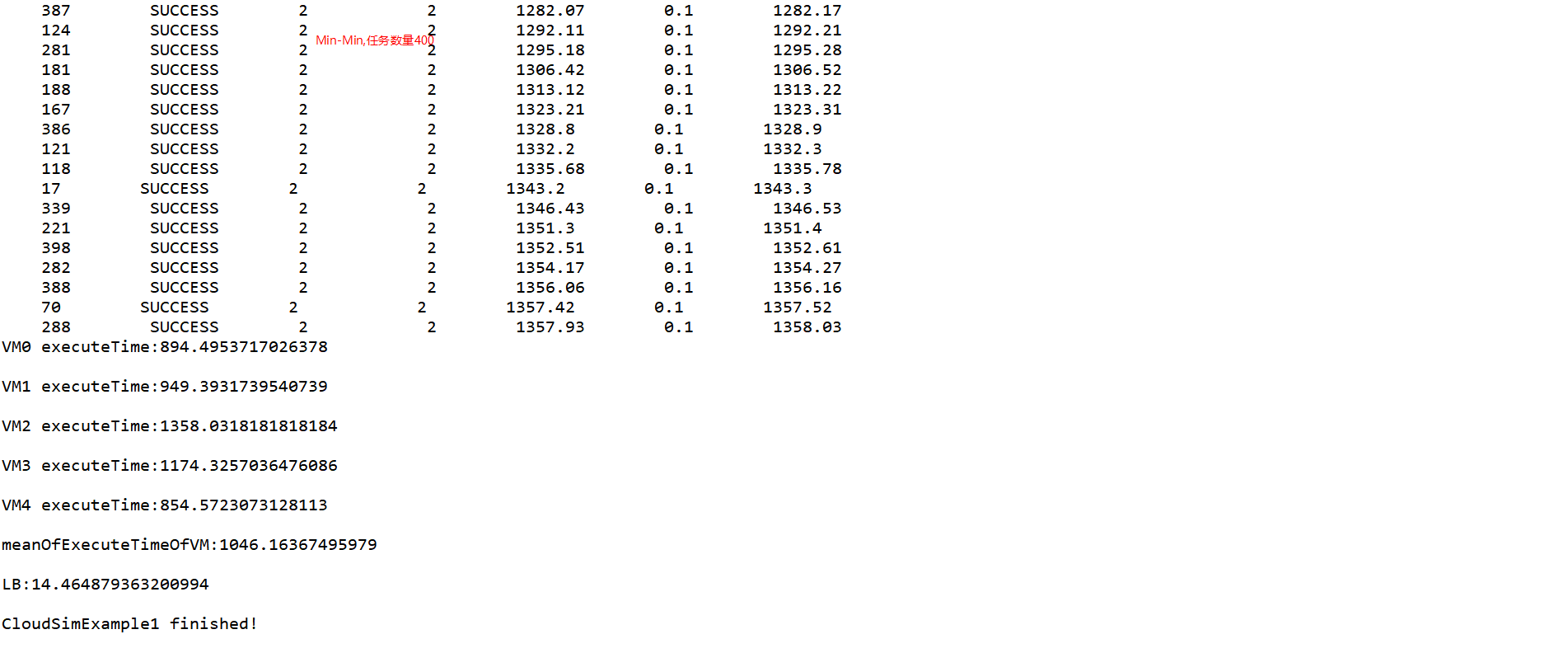
* 1. 遗传算法



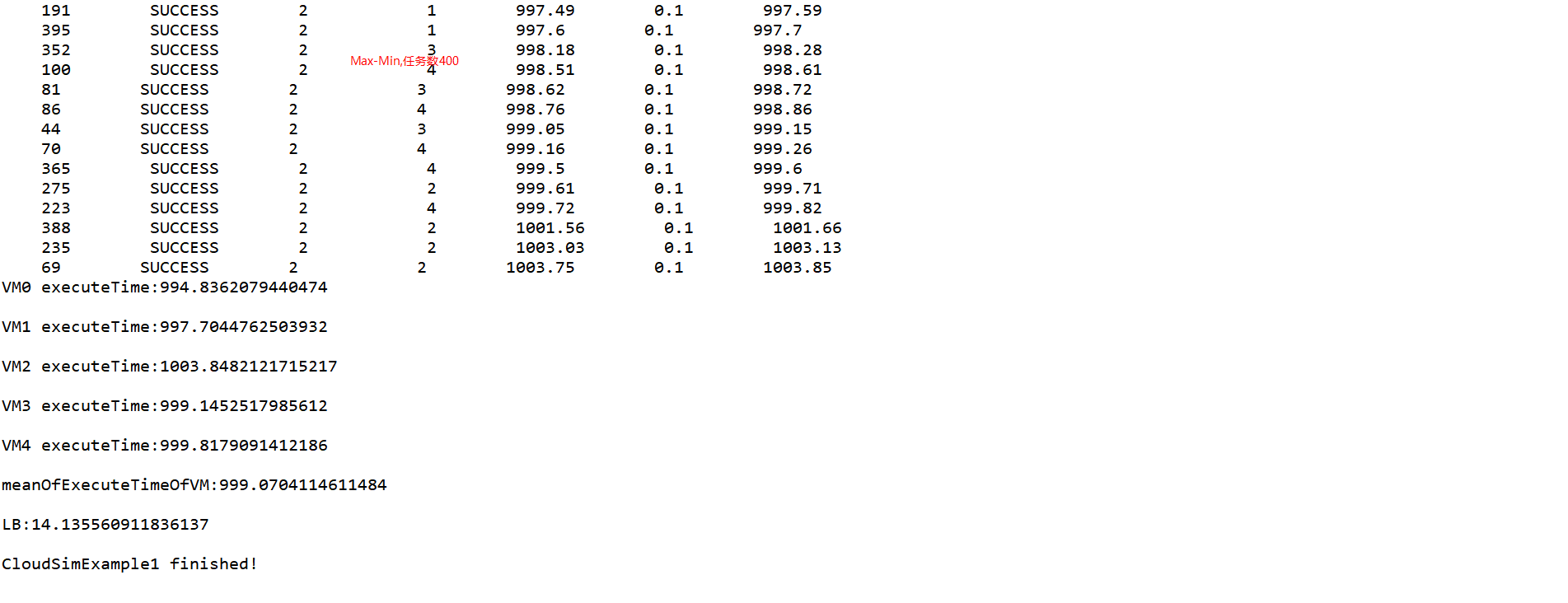
1. 任务数量400个
   1. 循环调度



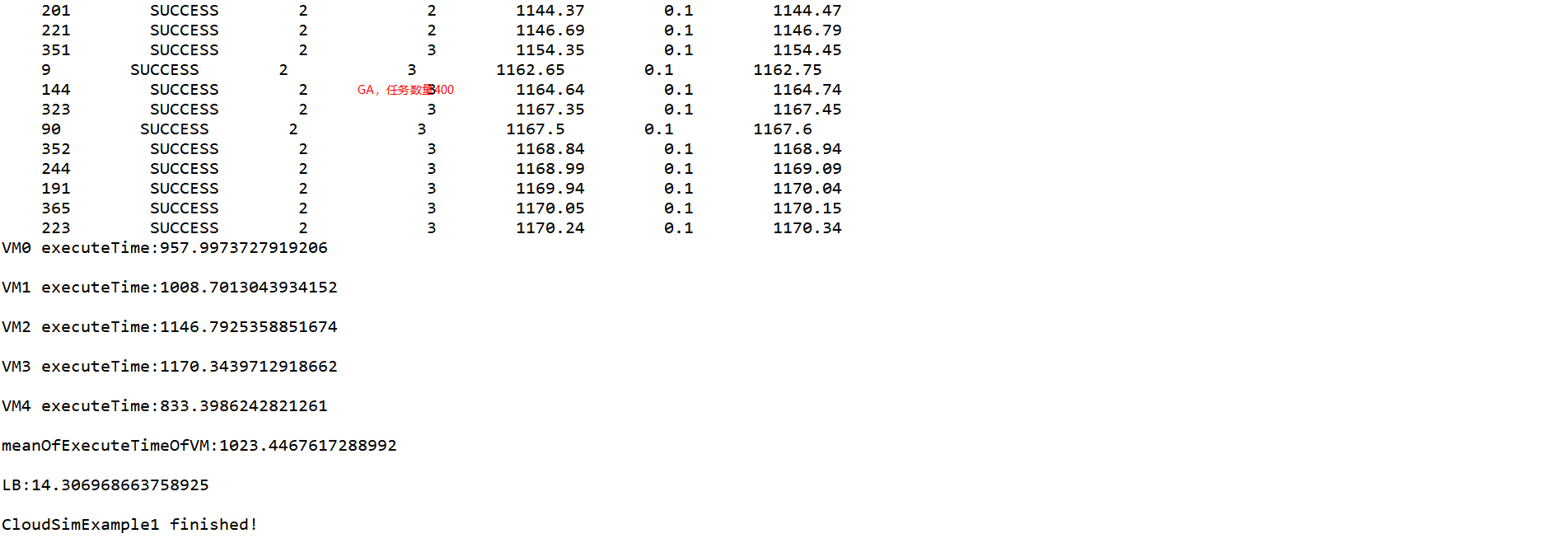
* 1. Min-min



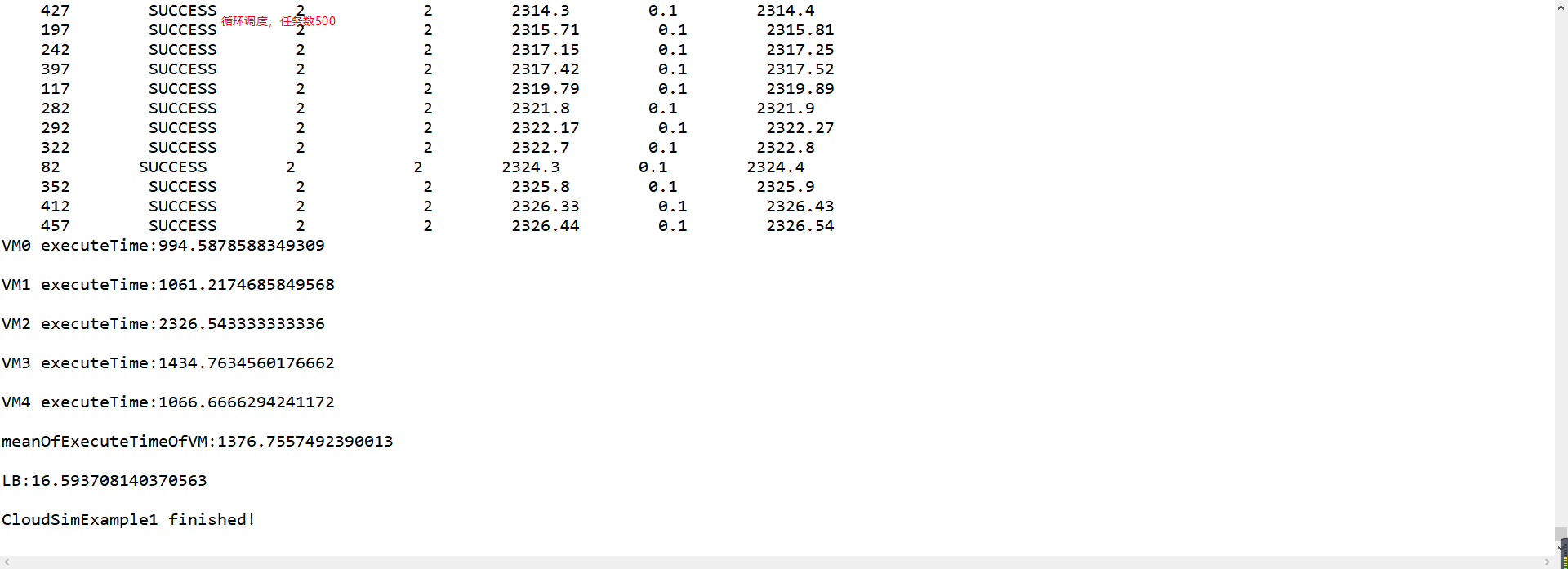
* 1. Max-min



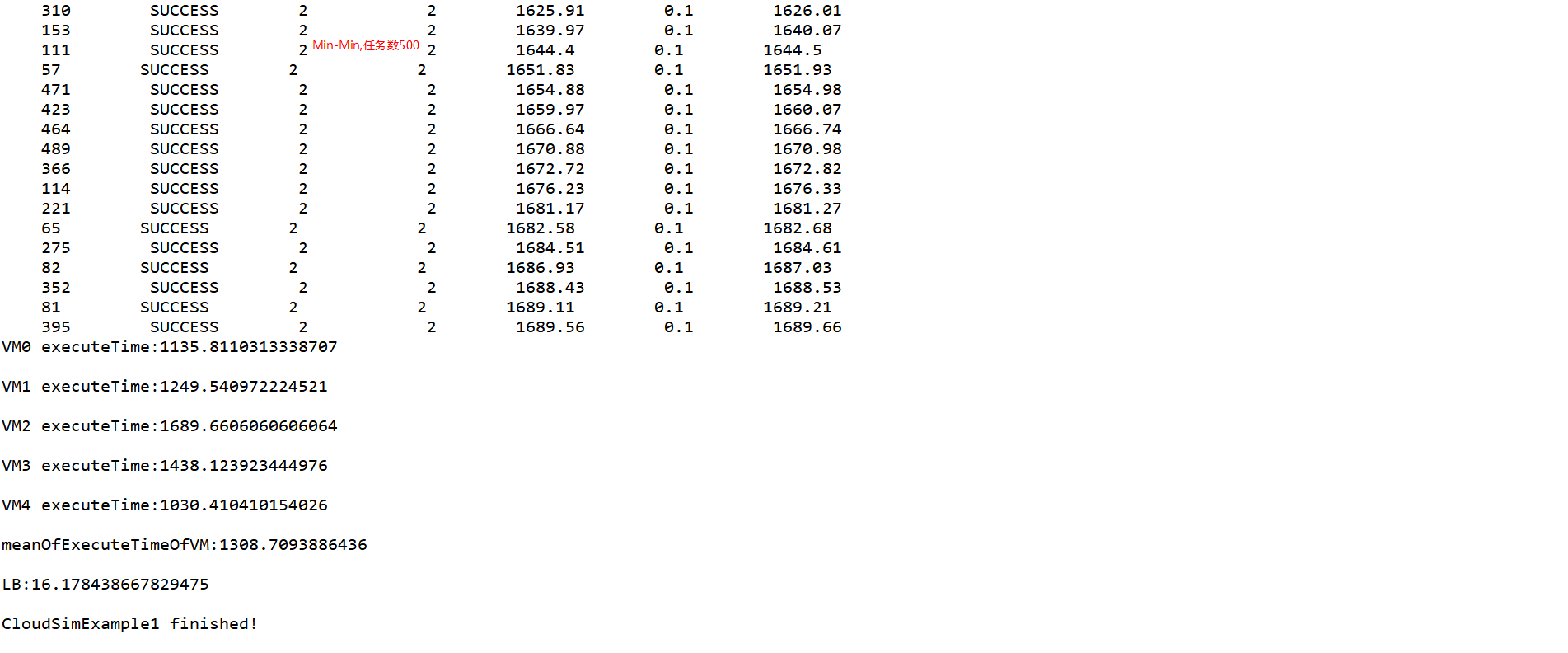
* 1. 遗传算法



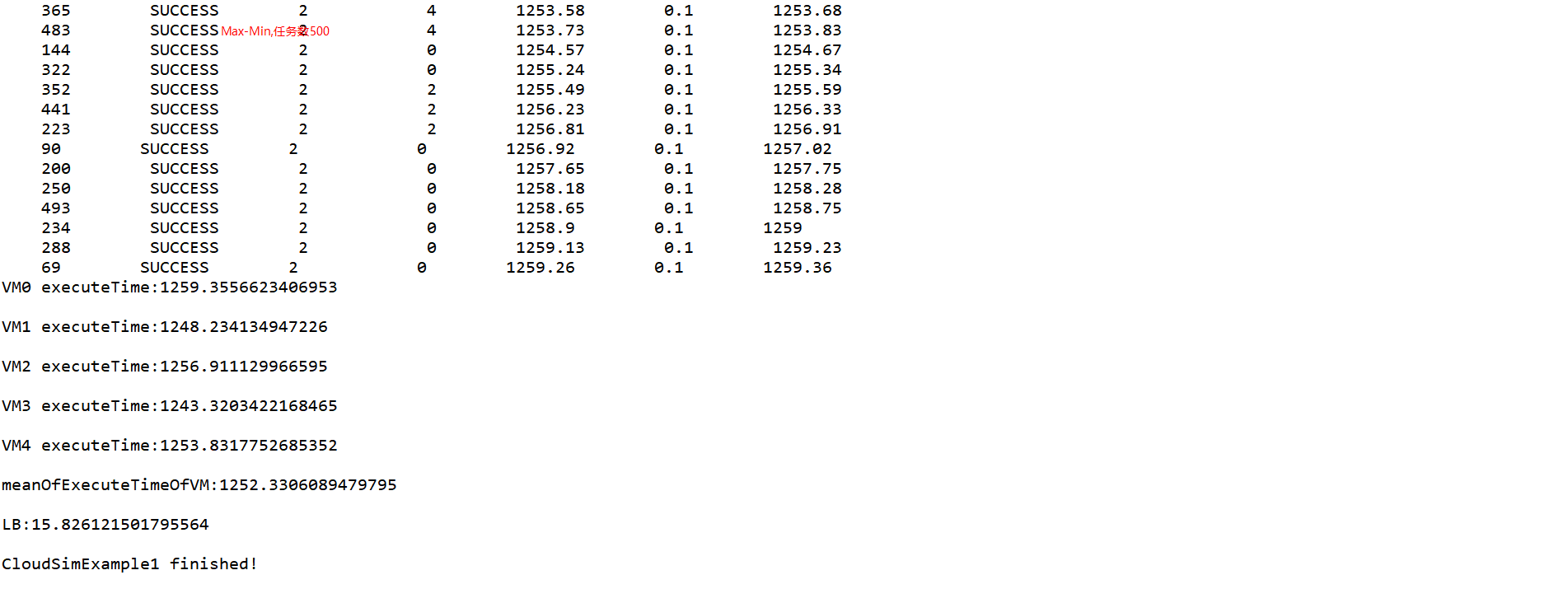
1. 任务数量500个
   1. 循环调度



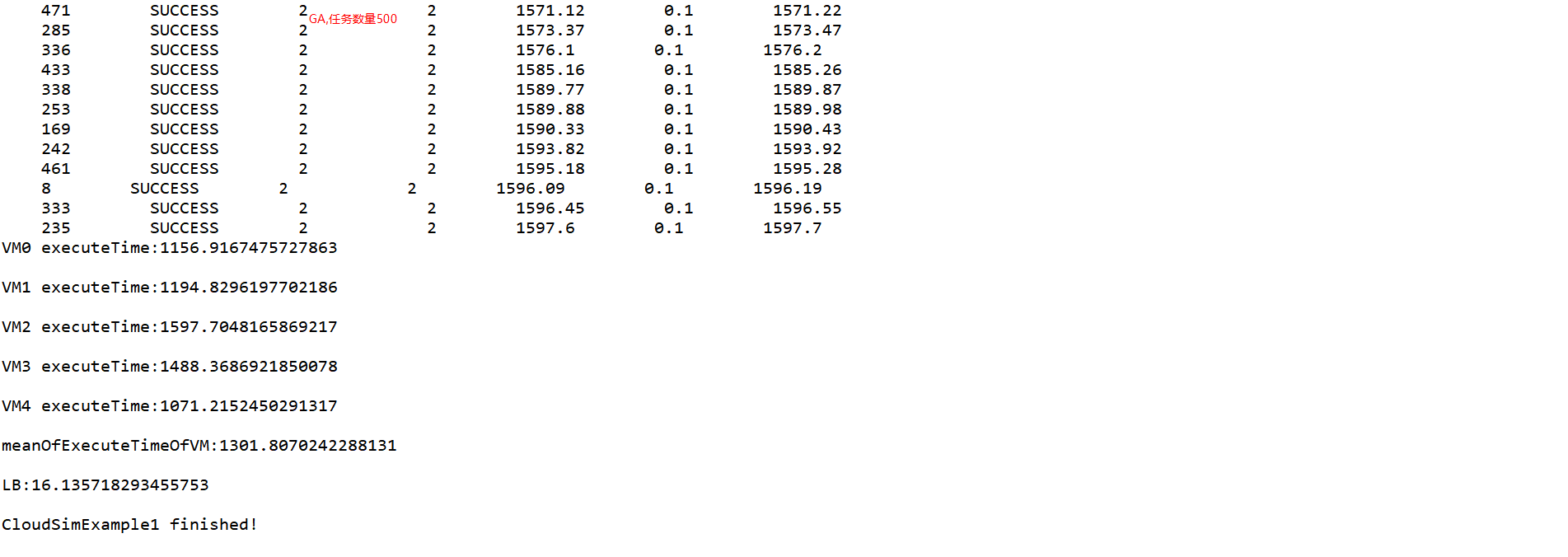
* 1. Min-min



* 1. Max-min



* 1. 遗传算法



### 绘制表格及图表

表 3 任务执行时间

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 策  略  任务  数 | 40 | 100 | 150 | 200 | 300 | 400 | 500 |
| 循环 | 1842.282 | 441.142 | 646.637 | 868.608 | 1330.327 | 1822.426 | 2326.543 |
| Min-Min | 1150.911 | 340.508 | 513.894 | 680.822 | 1042.729 | 1358.032 | 1689.661 |
| Max-min | 1139.437 | 246.314 | 376.064 | 488.641 | 744.306 | 1003.848 | 1259.356 |
| GA | 1191.208 | 289.124 | 514.796 | 609.695 | 847.348 | 1170.343 | 1597.705 |

**相应图表**



图 5 不同的任务数量在不同的算法调度情况下的执行时间

表 4 任务均衡度

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 策  略  任务  数 | 40 | 100 | 150 | 200 | 300 | 400 | 500 |
| 循环 | 15.146 | 7.188 | 8.965 | 10.274 | 12.706 | 14.794 | 16.594 |
| Min-Min | 14.569 | 7.039 | 8.822 | 10.082 | 12.480 | 14.465 | 16.178 |
| Max-min | 14.434 | 6.871 | 8.583 | 9.796 | 12.152 | 14.136 | 15.826 |
| GA | 14.534 | 6.969 | 8.811 | 10.024 | 12.295 | 14.307 | 16.135 |

**相应图表**



图 6 不同数量的任务在不同的算法调度下的均衡度

## 总结分析

通过实验结果可以看到使用遗传算法的任务调度执行时间在Min-min和Max-min之间，相对于CloudSim默认的遗传调度算法有了很大的提升，但Min-min和Max-min不可避免地存在饥饿现象。

## 参考文献

[1] 于彦波. 基于CloudSim平台的云资源调度策略研究[D/OL]. 华北电力大学(北京), 2017. https://kns.cnki.net/KCMS/detail/detail.aspx?dbcode=CMFD&dbname=CMFD201801&filename=1017222046.nh&v=.

[2] 张家铭. 基于粒子群算法的云计算资源调度优化研究[D/OL]. 华北电力大学(北京), 2019. https://kns.cnki.net/KCMS/detail/detail.aspx?dbcode=CMFD&dbname=CMFD202001&filename=1019237444.nh&v=.

[3] 张本志. 云计算中基于CloudSim的任务调度研究[D/OL]. 东北财经大学, 2019. https://kns.cnki.net/KCMS/detail/detail.aspx?dbcode=CMFD&dbname=CMFD202002&filename=1020033677.nh&v=.

[4] 罗肖辉, 徐美霞. 基于CloudSim云平台的任务调度与能耗分析研究[J]. 计算机与数字工程, 2019, 47(12): 3228-3234.

[5] 王鑫, 王人福, 蒋华. 改进贪婪算法的云任务调度研究[J/OL]. 微电子学与计算机, 2018, 35(2): 109-112+117. https://doi.org/10.19304/j.cnki.issn1000-7180.2018.02.023.

[6] 张焕青, 张学平, 王海涛, 等. 基于负载均衡蚁群优化算法的云计算任务调度[J/OL]. 微电子学与计算机, 2015, 32(5): 31-35+40. https://doi.org/10.19304/j.cnki.issn1000-7180.2015.05.007.

[7] 黄伟建, 辛风俊, 黄远. 基于混沌猫群算法的云计算多目标任务调度[J/OL]. 微电子学与计算机, 2019, 36(6): 55-59. https://doi.org/10.19304/j.cnki.issn1000-7180.2019.06.012.

[8] 黄璐. 基于遗传算法的云计算任务调度算法研究[D/OL]. 厦门大学, 2014. https://kns.cnki.net/KCMS/detail/detail.aspx?dbcode=CMFD&dbname=CMFD201402&filename=1014223219.nh&v=.

[9] 张浩荣, 陈平华, 熊建斌. 基于蚁群模拟退火算法的云环境任务调度[J]. 广东工业大学学报, 2014, 31(03): 77-82.