## CloudSimDPGA项目

### 简介

通过对CloudSim4.0进行改进，提供了评估DPGA算法策略的容器资源管理环境。可以实现容器的放置相关算法的对比。

### 模块介绍

* + 1. **Container核心模块**

ContainerHost：模拟主机，分配虚拟机资源，并更新虚拟机当前运行状态。

ContainerVM：模拟虚拟机。容器分配资源。

Container：模拟容器。负责给工作负载分配资源，并更新工作负载当前运行状态。一个容器运行单个负载。在仿真过程中，工作负载完成任务后，其所在容器重新初始化，作为下一个工作负载的运行空间。

ContainerCloudlet：模拟工作负载。

ContainerDatacenter:模拟数据中心，循环执行事件，模拟任务执行过程。

ContainerDatacenterBroker：模拟数据中心代理，负责给代替用户向用户数据中心提交任务请求。

* + 1. **DPGA模块**

该部分实现了GA和DPGA的调度如图1-1为DPGA的基本的代码结构图。

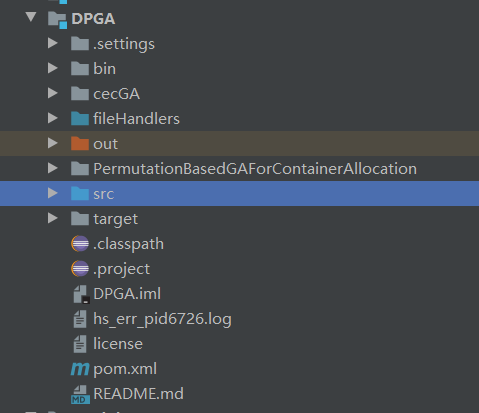


图1-1 DPGA源代码

Src为GA算法的基本框架，包括适应度计算，交叉、选择、变异，种群初始化等操作。PermutationBasedGAForContainerAllocation实现的DPGA算法。CecGA为传统遗传算法的实现。fileHandlers为数据文件和实验结果处理方法这里介绍DPGA的实现。

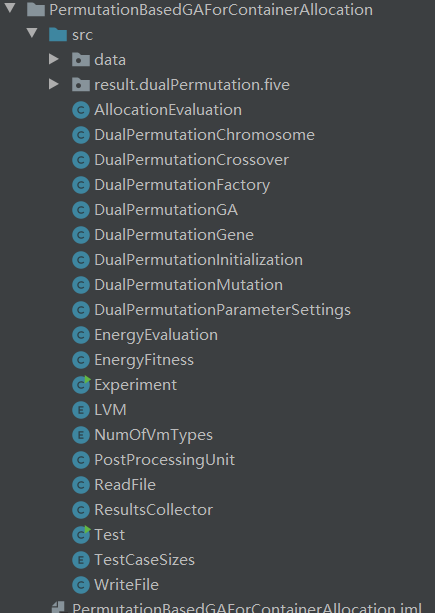


图1-2 DPGA实现

DualPermutationChrmosome实现了将虚拟机表、容器表、主机表用双排列染色体编码方法表示。这里实现了生成个体的交叉操作，以生成固定数量的种群。

DualPermutationCrossover实现了单点交叉和顺序交叉的操作。

DualPermutationMutation实现了单点变异和交换变异的方法。

EnergyEvaluation实现了种群能耗的评估，更新适应度值。

EnergyFitness实现染色体的解码，构建完整的解决方案，最后根据PmList的完整解决计算能耗。类中定义了虚拟机表和容器的映射方法以及物理机映射方法。

DualPermutationInitialization初始化种群，生成个体。生成容器随机序列即首先初始化具有容器长度的序列，然后随机散列。

DualPermutationGene实现了染色体基因，实现了复制和获取相应排列的操作。

DualPermutationParameterSettings获取实验中容器、虚拟机和主机的资源定义，获取使用的交叉、变异操作。

DualPermutationFactory获取在遗传算法中使用的初始化、变异、选择、交叉、排序和数据收集器的类型。

PostProcessingUnit定义了输出结果的计算，包括能耗、平均能耗、平均主机数、虚拟机数、平均物理机的CPU和内存利用率、以及计算N次运行后的收敛曲线，并且将这些数据输出到文件中。

Experiment设置实验环境和数据，如图1-3初始化集群环境，设置了虚拟机类型和容器数量。此外设置了遗传算法的种群大小、迭代次数、交叉概率和变异概率。之后执行DPGA算法的相关操作，输出结果。

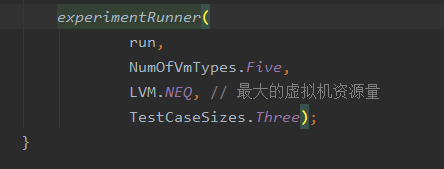


图1-3 实验环境设置

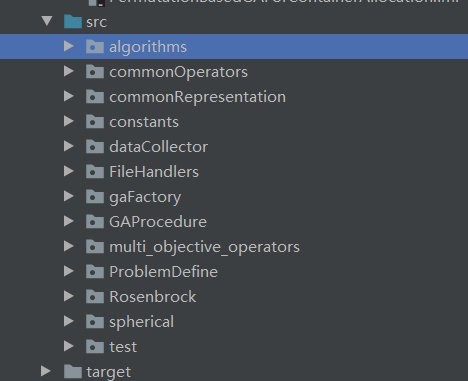


图1-3 GA框架

在src中为GA算法的基本框架，改进的算法需要继承里面的类。这里包括了相应的交叉、变异、数据收集器以及染色体表示的抽象类。

* + 1. **ContainerVmProvisioners模块**

该模块主要负责CPU、内存、网络带宽资源向虚拟机供给的过程。

ContainerVmPeProvisioner：模拟CPU资源供给，不允许超量使用CPU资源。

ContainerVmRamProvisioner模拟内存资源供给。

* + 1. **容器资源供给模块containerProvisioners**

该模块主要负责模拟CPU、内存、网络带宽资源向容器供给的过程。

ContainerPeProvisioner：模拟CPU资源供给，不允许超量使用CPU资源；

ContainerRamProvisioner：模拟内存资源供给；

ContainerBwProvisioner：模拟网络带宽资源供给。

* + 1. **调度模块schedulers**

模拟虚拟机、容器、工作负载进行资源分配和管理。

ContainerVmScheduler：抽象类，负责虚拟机资源管理和分配。

ContainerScheduler：抽象类，负责容器资源管理和分配。

ContainerVmSchedulerTimeShared：基于时间共享但不允许CPU超量使用的资源管理方案。

ContainerSchedulerTimeShared：基于时间共享但不允许CPU超量使用的资源管理方案。

ContainerCloudletScheduler：抽象类，负责工作负载的资源管理和分配；

ContainerCloudletSchedulerTimeShared：基于时间共享的资源管理方案；

ContainerCloudletSchedulerDynamicWorkload：支持工作负载动态变化的基于时间共享的资源管理方案。

* + 1. **ContainerPlacementPolocies**

主要为容器的放置，选择合适的虚拟机进行放置。

containerPlacementPolicies抽象类，自定义实现的具体策略序继承该类或该类的子类。

ContainerPlacementPolicyFirstFit实现了FirstFit算法。

ContainerPlacementPolicyMostFull实现了MostFull算法。

* + 1. **resourceAllocators**

### 实验数据

使用workload内的容器数据进行仿真实验，ReadFile为读取相关文件的代码。Workload保存了容器的负载数据，还有虚拟机和主机的配置文件。

### 运行说明

* + 1. **执行FirstFit、MostFull**

1. org.cloudbus.cloudsim.examples.container,ContainerExamples设置相应的虚拟机主机、容器的数量。
2. 创建相应的容器放置策略，在本实验创建的ContainerInitialPlacementTest实现了FirstFit方法。
3. 执行程序cloudsim-example.container.ContainerInitialPlacementTest。在控制台显示了运行的时间戳，在日志区域输出相应的输出文件。日志记录了能耗、活跃主机数、运行日志等。

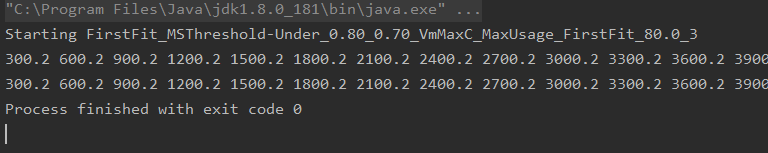


图1-4 输出

在~/result可以查看输出文件

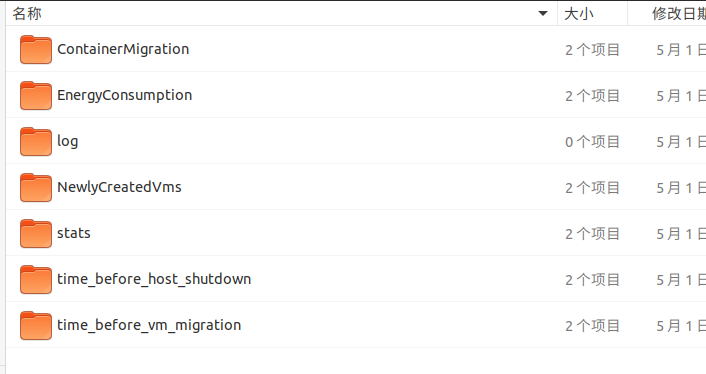


图1-5 记录文件

在EnergyConsumption给出了能耗数据，在stats文件给出了各项的数据，如下图给出SLA数据：

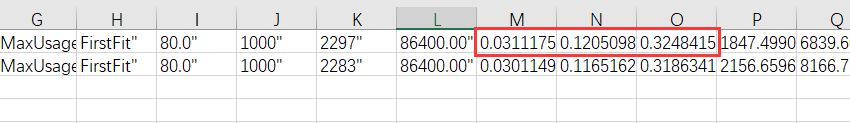


图1-6 SLA违反率

* + 1. **在IDEA上执行DPGA**

1. 使用Maven将CloudSimDPGA导入IDEA，在DPGA.PermutationBasedGAForContainerAllocation.Experiment进行实验环境的设置。

在IDEA菜单栏File->Project Structure->modules选中DPGA，选择Sources，增加source folders，将PermutationBasedGAForContainerAllocation的src和DPGA下的src标记为Sources，将out、target设置为Excluded，如下图：

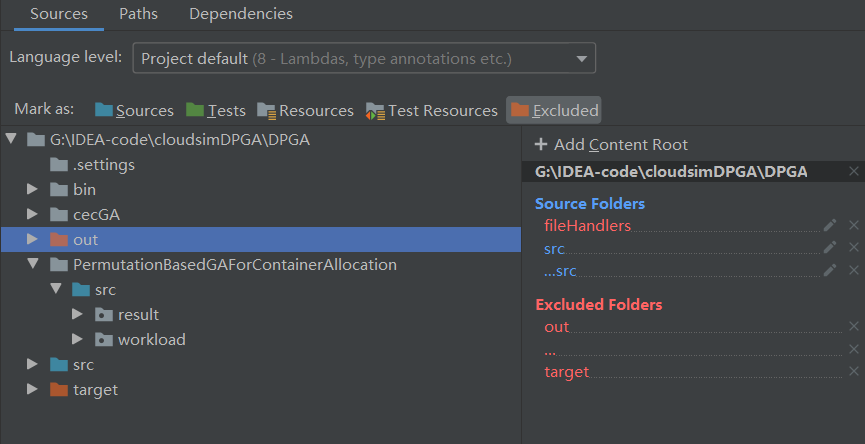


图1-7 文件配置

1. 运行Experiment类。执行完之后在控制台看到相应的能耗能耗、每次运行的平均能耗、虚拟机数量、主机数量、CPU、内存的利用率等。在相应的日志文件也可以看到实验结果。

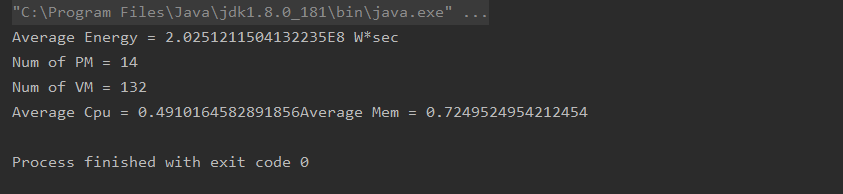


图1-8 DPGA输出

输出结果文件如下图：

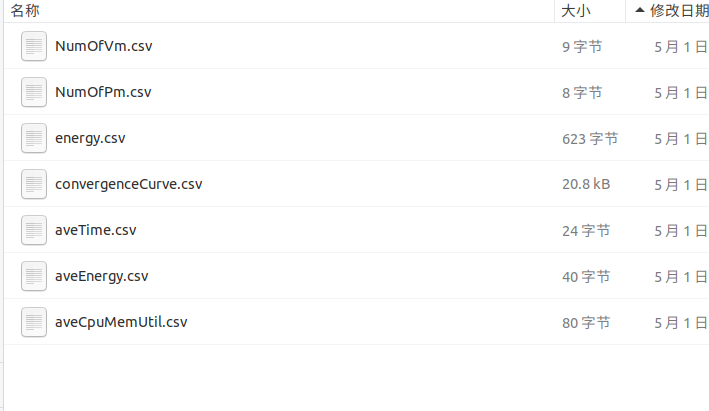


图1-9 DPGA结果

1. 执行GA算法，在src/cecGA文件下运行Experiment.java文件即可。