**获奖赛队名称：sukelyc\_xxp**

**所属赛题：华为算法精英实战营第八期 - JSP调度和内存背包问题**

**赛题核心点介绍：**1. 任务依赖与资源调度的优化：合理安排任务的执行顺序和资源分配，确保任务之间的依赖关系不会影响整体的调度效率。

2. 硬盘与服务器资源的高效利用：根据硬盘容量和服务器的处理能力，尽可能优化存储与计算的分配，减少等待时间。

3. 避免资源冲突：确保多个任务在同一资源上运行时不会产生冲突，最大化利用硬件性能。

**整体思路介绍：**

1. 初始分配：根据任务依赖关系和可用资源，选择最优的服务器和硬盘进行任务分配。
2. 排产优化：使用贪心策略获得初步的任务顺序，优先选择能最早完成任务的资源。
3. 局部搜索：通过 2-opt、3-opt、Relocate、Exchange 等算子，调整任务顺序进行优化。
4. 终止条件：在设定时间内或无显著改善时停止优化

**题目理解:**

随着云计算、5G通信、人工智能等芯片硬件计算能力的快速发展，硬件架构正向更复杂的多核、异构架构演进。作业调度与内存分配的耦合问题广泛存在于操作系统、分布式系统和编译器中，在软硬件协同优化中起着关键作用。在该问题中，需要设计高效的服务器调度算法和硬盘分配策略。即需要将计算任务分配到不同的服务器上，除此之外，针对这些任务需要从可用的硬盘中分配一块硬盘用于存储数据。

其中任务数量为，机器数量为，机器的处理速度为，硬盘数量为，硬盘的读写速率为，存储容量为。针对每一个任务,我们可以假设其可以在时间点在服务器上进行处理，同时数据存储到硬盘上。

服务器的处理速度为 ，针对每个任务，都有任务大小, 那么任务在服务器上的处理时间为。

硬盘的读写速率为 ，针对每个任务，都有数据大小, 那么任务在服务器上的处理时间为。

在任务执行前，可能需要读取其他任务的数据才能执行，我们定义为任务执行前需要读取的任务集合，那么这个读取数据所需的执行时间为 .

综上所述，当任务启动时，它会经历三个阶段:从其他任务读取数据、执行任务、存储数据。为方便起见，我们将这三个阶段的开始时间记为，最后一个阶段的结束时间记为。这些值的确定方法如下。

* .
* .
* .

此外，可行的任务调度还应满足以下要求:

无抢占:每台机器一次只执行一个任务或传输一个数据，在其生命周期内不允许中断。换句话说，对的每一对任务和，两个间隔和不能有任何重叠。

任务依赖关系:每个任务都有一个与任务相关的任务列表，用表示。对于每个任务，任务在完成执行之前不能启动。也就是说，它必须满足.注意，如果任务和被安排在同一台机器上，那么它仍然需要等待完成对其数据的存储。

数据依赖性:对于每个任务，任务只能在存储完数据后才能启动。也就是说，它必须满足。

机器的可用性:每个任务都有一个可用机器的列表。用表示，每个任务都必须被安排在与它的可用机器上。也就是说，它必须满足。

硬盘容量:每个硬盘的存储容量为。同一硬盘上存储的所有数据的总大小不能超过该硬盘的容量。也就是说，对于每个硬盘它必须满足下式

**核心策略详解**

1. 记当前时刻
2. 获取当前系统中所有服务器的最后一笔任务的完成时间，记为时刻，取所有满足的设备可用设备
3. 获取当前系统中满足任务依赖性，数据依赖性，且任务对应的可用设备当前可用的

任务作为可安排的任务。若无可用任务，则并返回第二步。

1. 遍历所有可安排任务和对应当前可用设备可得的一系列排列组合预求解结果，取最小对应的结果，即任务在时间点在服务器上进行处理,同时获取当前情况下满足的处理速度最快的硬盘作为处理的硬盘。即获取当前可用最早能完成加工的任务并将其分配到可用的处理最快的硬盘上
2. 更新排产结果，若所有任务排产完毕，则停止，否则返回第二步。
3. 现我们可以得到一个长度为任务的加工顺序序列，我们针对这个加工序列执行以下领域搜索,若领域搜素得到的解更优秀，则返回新解，直到求解时间到达求解时长上限(如15s),或连续n次领域搜索解无变化.

A1, 2-opt 选定两个点，将这两个点之间的区间反转, 如

选择e, c可得(..., e, a, c, b, d, ...)

选择e, a可得(..., e, b, c, a, d, ...)

选择e, b可得(..., e, d, a, c, b, ...)

选择d, a可得(..., e, d, c, a, b, ...)

选择d, c可得(..., e, d, b, a, c, ...)

选择b, a可得(..., e, d, b, c, a, ...)

A2, 2-h-opt 随机产生两点，塞进新排列头部。其余的按顺序往后逐个塞进去, 如

选择e, d可得(..., e, b, c, a, d, ...)

选择d, e可得(..., d, b, c, a, e, ...)

选择e, b可得(..., e, d, c, a, b, ...)

选择b, e可得(..., b, d, c, a, e, ...)

选择e, c可得(..., e, d, b, a, c, ...)

选择c, e可得(..., c, d, b, a, e, ...)

选择e, a可得(..., e, d, b, c, a, ...)

选择a, e可得(..., a, d, b, c, e, ...)

选择d, b可得(..., d, e, c, a, b, ...)

选择b, d可得(..., b, e, c, a, d, ...)

选择d, c可得(..., d, e, b, a, c, ...)

选择c, d可得(..., c, e, b, a, d, ...)

选择d, a可得(..., d, e, b, c, a, ...)

选择a, d可得(..., a, e, b, c, d, ...)

选择b, c可得(..., b, e, d, a, c, ...)

选择c, b可得(..., c, e, d, a, b, ...)

选择b, a可得(..., b, e, d, c, a, ...)

选择a, b可得(..., a, e, d, c, b, ...)

选择c, a可得(..., c, e, d, b, a, ...)

选择a, c可得(..., a, e, d, b, c, ...)

A3, 3-opt 通过选取路径中不相邻的三个节点之间的连接删除，然后尝试其他不同连接方式, 如

选择e, d, b可得(..., e, b, d, c, a, ...)

选择e, d, b可得(..., e, b, d, c, a, ...)

选择e, d, b可得(..., e, b, d, c, a, ...)

选择e, d, b可得(..., e, b, d, c, a, ...)

选择e, d, c可得(..., e, c, b, d, a, ...)

选择e, d, c可得(..., e, b, c, d, a, ...)

选择e, d, c可得(..., e, b, c, d, a, ...)

选择e, d, c可得(..., e, b, c, d, a, ...)

选择e, d, a可得(..., e, a, c, b, d, ...)

选择e, d, a可得(..., e, b, c, a, d, ...)

选择e, d, a可得(..., e, b, c, a, d, ...)

选择e, d, a可得(..., e, b, c, a, d, ...)

选择e, b, c可得(..., e, c, b, d, a, ...)

选择e, b, c可得(..., e, c, d, b, a, ...)

选择e, b, c可得(..., e, c, b, d, a, ...)

选择e, b, c可得(..., e, c, b, d, a, ...)

选择e, b, a可得(..., e, a, c, b, d, ...)

选择e, b, a可得(..., e, c, a, d, b, ...)

选择e, b, a可得(..., e, c, a, b, d, ...)

选择e, b, a可得(..., e, c, a, b, d, ...)

选择e, c, a可得(..., e, a, c, b, d, ...)

选择e, c, a可得(..., e, a, d, b, c, ...)

选择e, c, a可得(..., e, a, c, b, d, ...)

选择e, c, a可得(..., e, a, c, b, d, ...)

选择d, b, c可得(..., e, d, c, b, a, ...)

选择d, b, c可得(..., e, d, c, b, a, ...)

选择d, b, c可得(..., e, d, c, b, a, ...)

选择d, b, c可得(..., e, d, c, b, a, ...)

选择d, b, a可得(..., e, d, a, c, b, ...)

选择d, b, a可得(..., e, d, c, a, b, ...)

选择d, b, a可得(..., e, d, c, a, b, ...)

选择d, b, a可得(..., e, d, c, a, b, ...)

选择d, c, a可得(..., e, d, a, c, b, ...)

选择d, c, a可得(..., e, d, a, b, c, ...)

选择d, c, a可得(..., e, d, a, c, b, ...)

选择d, c, a可得(..., e, d, a, c, b, ...)

选择b, c, a可得(..., e, d, b, a, c, ...)

选择b, c, a可得(..., e, d, b, a, c, ...)

选择b, c, a可得(..., e, d, b, a, c, ...)

选择b, c, a可得(..., e, d, b, a, c, ...)

A4 4-OPT 通过选取路径中不相邻的4个节点之间的连接删除，然后尝试其他不同连接方式,

选择e, d, b, c可得(..., e, c, b, d, a, ...)

选择e, d, b, a可得(..., e, c, a, b, d, ...)

选择e, d, c, a可得(..., e, a, b, c, d, ...)

选择e, b, c, a可得(..., e, a, c, d, b, ...)

选择d, b, c, a可得(..., e, d, a, c, b, ...)

A5, Relocate算子也被称为移动算子，它可以将一个节点从当前路径中移动到另一个位置。与Or-opt算子类似，但Relocate算子只移动一个节点，而不涉及路径的重新连接，如

选择a放置在b前可得(..., a, b, c, d, e, ...)

选择a放置在c前可得(..., b, a, c, d, e, ...)

选择a放置在d前可得(..., b, c, a, d, e, ...)

选择a放置在e前可得(..., b, c, d, a, e, ...)

选择b放置在c前可得(..., a, b, c, d, e, ...)

选择b放置在d前可得(..., a, c, b, d, e, ...)

选择b放置在e前可得(..., a, c, d, b, e, ...)

选择c放置在a前可得(..., c, a, b, d, e, ...)

选择c放置在d前可得(..., a, b, c, d, e, ...)

选择c放置在e前可得(..., a, b, d, c, e, ...)

选择d放置在a前可得(..., d, a, b, c, e, ...)

选择d放置在b前可得(..., a, d, b, c, e, ...)

选择d放置在e前可得(..., a, b, c, d, e, ...)

选择e放置在a前可得(..., e, a, b, c, d, ...)

选择e放置在b前可得(..., a, e, b, c, d, ...)

选择e放置在c前可得(..., a, b, e, c, d, ...)

A6, Exchange算子：Exchange算子是一种交换操作，用于交换两个节点之间的位置。通过选择两个节点，并交换它们在路径中的位置，可以改变节点的顺序。

交换a和b的位置可得(..., b, a, c, d, e, ...)

交换a和c的位置可得(..., c, b, a, d, e, ...)

交换a和d的位置可得(..., d, b, c, a, e, ...)

交换a和e的位置可得(..., e, b, c, d, a, ...)

交换b和c的位置可得(..., a, c, b, d, e, ...)

交换b和d的位置可得(..., a, d, c, b, e, ...)

交换b和e的位置可得(..., a, e, c, d, b, ...)

交换c和d的位置可得(..., a, b, d, c, e, ...)

交换c和e的位置可得(..., a, b, e, d, c, ...)

交换d和e的位置可得(..., a, b, c, e, d, ...)