

# 打包调度机制

gumblx

## 1 定义

分布式打包调度首先需要解析依赖，对依赖做拓扑排序，然后把同级的依赖包分配到不同机器去完成编译。其中，依赖解析和拓扑排序有较为通用的算法。我们在此解决的问题主要是如何去将包分配给各台不同的机器。

设包  $i$  的工作量为  $w_i$ 。设某特定软件包（如 *glibc*）的工作量为 1。

对于机器  $j$ ，设其编译速度（编译单位工作量所需时间）为  $v_j$ 、CPU 个数为  $c_j$ ，且有内存限制  $M_j$ 、硬盘限制  $D_j$ 。

用机器  $j$  编译包  $i$  时，可以观察到实际使用时间（real） $t_{ij}$ 、CPU 使用时间（user+sys） $T_{ij}$ ，以及最大内存使用量  $m_i$  和最大硬盘使用量  $d_i$ 。相同架构的机器编译所使用的内存和硬盘基本相同。

## 2 参数估计

### 工作量和性能

用机器  $j$  编译包  $i$  时，有

$$\frac{w_i}{v_j} = T_{ij} \Rightarrow \log(w_i) - \log(v_j) = \log(T_{ij})$$

列出所有机器编译每个包所需 CPU 时间的数值，以  $\log(w_i)$  和  $\log(v_j)$  作为未知数，建立稀疏矩阵，可使用最小二乘法求解该线性系统，得到每个  $w_i$  和  $v_j$  的估计值。

### 并行效率

定义包  $i$  在  $c$  核机器上编译的并行效率为

$$P_i(c) = \max\{0, \frac{T_i/t_i - 1}{c - 1}\}$$

实际编译时间即为

$$t_{ij} = \frac{w_i}{v_j(1 + (c - 1) \cdot P_i(c))} \quad (1)$$

假设  $P_i(c)$  为一次函数  $k_i c + b_i$ ，用最小二乘法线性回归可求出参数  $k_i$  和  $b_i$ 。

若存在  $x$  使  $P_i(x) = 0$ ，则认为函数即为  $P_i(c) = 0$ ，表示编译过程为单线程。

### 3 工作分配

使用上述方法可以估算出每个包的工作量  $w_i$  (默认为 1)、包的并行效率  $P_i(c)$  (默认参数为所有包的平均)、每台机器的效率  $v_j$  (需要进行基准测试)。用式 (??) 可估算出包  $i$  在机器  $j$  上编译的时间  $t_{ij}$ 。

设目标函数  $z$  为所需最大时长。决策变量  $a_{ij}$  为是否将包  $i$  分配给机器  $j$ 。总共有  $X$  个包、 $Y$  台机器。

列出整数线性规划问题：  $\min z$

$$\text{s.t.} \begin{cases} \sum_{j=1}^Y a_{ij} = 1, & i = 1, 2, \dots, X & (2a) \\ z \geq \sum_{i=1}^X a_{ij} t_{ij}, & j = 1, 2, \dots, Y & (2b) \\ a_{ij} m_i \leq M_j & & (2c) \\ a_{ij} d_i \leq D_j & & (2d) \\ a_{ij} \in \{0, 1\} & & (2e) \end{cases}$$

其中，约束条件 (??a) 表示同一个包只能分配给一台机器；(??b) 表示目标函数最大时长要大于每台机器上所有任务的时长之和；(??c)、(??d) 表示不把任务分配给不满足内存和硬盘要求的机器；(??e) 表示决策变量  $a_{ij}$  为 0-1 变量。