吴雨娟 23920192204097

厦門大學
XIAMEN UNIVERSITY
ADD:FUJIAN XIAMEN CABLE:0633 P.C:361005

12.7

解: 16KB = 16 × 2^{10}B = 16384B

① 41600 ÷ 16384 = 2块 … 8832B

$\frac{8832}{16384} \times 100\% \approx 53.9\%$

② 640000 ÷ 16384 = 39块 … 1024B

$\frac{1024}{16384} \times 100\% = 6.25\%$

③ 4064000 ÷ 16384 = 248块 … 768B

$\frac{768}{16384} \times 100\% \approx 4.69\%$

12.10

解: 这种限制对用户组织文件空间造成一些不好的影响. 假设操作系统构造一个文件系统以便子目录被允许包含在一个主目录底下, 但没有额外的思维被要求允许包含任意深度的子目录.

16.1

解: a. MIPS表达式: $(n\alpha+1-\alpha)T$

b. $(16\alpha+1-\alpha)\times 4 = 40$

$\alpha = 0.6$

16.2

解: a. 假设在集群上执行程序占用3时间T. 其中0.25T是占用程序在1台计算机上运行的时间. 如果把这0.25T时间内1台计算机执行完的代码交给一台计算机执行. 所需的时间是 0.25T × 9 = 2.25T. ∴ 在一台计算机上执行程序占用的时间是 2.25T + 0.75T = 3T. ∴ 有效加速比为 $\frac{3T}{T} = 3$. $\alpha = \frac{2.25T}{3T} = 0.75$.

b. 若能够在并行代码部分上有效地使用18台计算机，那么0.25T时间内18台计算机执行完的代码交给一台计算机执行所需的时间是 $0.25T \times 18 = 4.5T$ ∴在一台计算机上执行程序所用的时间是 $4.5T + 0.75T = 5.25T$. ∴有效加速比 $= \frac{5.25T}{T} = 5.25$

15.3

解：有。如果Pi已经发出一个请求消息但还没有收到相应的释放信息，Pi可以节省它自己的一个Reply消息的传输。

15.4

解：a. 证明：如果一个进程Pi将请求Request发送给其他所有进程，并收到了来自其他所有进程的Reply消息，说明这个进程Pi发出的请求是所有正在等待的请求中最早发出的，比这个请求更早的请求已被完成。假设有Pj发出一个更早的请求或它已经进入了临界区，那么Pj不会给Pi发送Reply消息，Pi就无法收到其他所有进程的Replay。矛盾。所以假设不成立。所以所有的请求会被排序，不存在有两个进程同时处于临界区的情况，所以它实现了互斥。

b. 由于每个Request都是排了序的，对Request的服务是按顺序的，每个Request都将在某段时间后成为最旧的，因此总会轮到服务，所以不可能会饥饿。

15.5

解：不一样。在令牌传递互斥算法中不考虑关于相互之间的事件时间戳时钟，时钟只用于被更新。request们通过请求消息的通信方式在其他进程中更新。当令牌在进程间传递时，token们会发生更新。所以时间戳的作用是计数器，记录多个进程请求临界区的时间戳，判断下一个接收令牌的进程是谁。

15.6

解: a. 证明: token_present; 是Pi的布尔值. 当Pi收到令牌后, token_present; 由false变成true. 只有当token_present; 将值变为true且Pi在令牌送出去之前把这个值变成false的时候. Pi才能把令牌送出去. 可以在任何情况下把值为true的token_present的值不超过1, 保证互斥的实现.

b. 证明: 假设所有进程都希望进入临界区, 但他们都没有令牌, 都处于阻塞状态. 此时令牌是处于被传出的状态, 在经过有限长的时间后, 令牌会被传给某一个进程, 再被传给进程. 所以该算法不会造成死锁.

c. 证明: 令牌总是会在经过有限长的时间后传给下一个进程. 所以对于所有想要进入临界区的进程, 总会轮到它拿到令牌, 进入临界区. 所以该算法是公平的.