HW1

王嵘晟

PB17111614

1. Server farms such as Google and Yahoo! provide enough compute capacity for the highest request rate of the day. Imagine that most of the time these servers operate at only 60% capacity. Assume further that the power does not scale linearly with the load; that is, when the servers are operating at 60% capacity, they consume 90% of maximum power. The servers could be turned off, but they would take too long to restart in response to more load. A new system has been proposed that allows for a quick restart but requires 20% of the maximum power while in this “barely alive” state.
2. How much power savings would be achieved by turning off 60% of the servers?

设一台服务器最大功率为P，共N台服务器。这时总功率NP。关掉60%服务器后，总功率0.4\*N\*P=0.4NP。所以共节约了0.6NP，即60%的功率

1. How much power savings would be achieved by placing 60% of the servers in the “barely alive” state?

设一台服务器最大功率为P，共N台服务器。这时总功率NP。将60%服务器置于“勉强生存”状态，总功率0.6\*N\*0.2\*P+0.4\*N\*P=0.52NP。所以共节约了0.48NP，48%的功率

1. How much power savings would be achieved by reducing the voltage by 20% and frequency by 40%?

能耗为原来的0.8\*0.8=0.64，所以功率为原来的0.64\*0.6=0.384

1. How much power savings would be achieved by placing 30% of the servers in the “barely alive” state and 30% off?

原总功率为NP，关掉30%，30%“勉强生存”后，总功率0.3\*N\*0.2P+0.4\*N\*P=0.46NP,所以节省了54%的功率

1. 公司刚刚买了一个新的 Intel Core i5 双核处理器， 你接到针对这一处理器来优化软件的任务。 你将在这个处理器上运行两个应用程序， 但它们的 资源需求并不一样。第一个程序需要 80% 的资源， 另一个需要 20% 的资源。假定对该程序的一部分进行并行化时， 该部分的加速比为 2.
   1. 假定第一个应用程序的 40% 可以并行化， 那么在隔离运行时， 通过这个应用程序可以实现多大的加速比？

1/(0.6+0.4/2)\*100%=125%

* 1. 假定第二个应用程序的 99% 可以并行化， 那么在隔离运行时， 通过这个应用程序可以实现多大的加速比？

1/(0.01+0.99/2)\*100%=198%

* 1. 假定第一个应用程序的 40% 可以并行化， 如果对其实现并行化， 系统总加速比为多少？

1/(0.2+0.8\*0.6+0.8\*0.4/2)\*100%=119%

* 1. 假定第二个应用程序的 99% 可以并行化， 如果对其实现并行化， 系统总加速比为多少？

1/(0.8+0.2\*0.01+0.2\*0.99/2)\*100%=111%

1. 假设你的 load/store 计算机具有一下指令组合：

|  |  |  |
| --- | --- | --- |
| Operation | Frequency | No.of Clock Cycle |
| ALU ops | 35% | 1 |
| Load | 25% | 2 |
| Store | 15% | 2 |
| Branches | 25% | 3 |

* 1. 计算CPI

CPI=35%\*1+25%\*2+15%\*2+25%\*3=1.9

* 1. 我们观察到 35% 的 ALU 操作都伴随着一条 load 指令。现在我们用一条新的指令来替代这些 ALU 操作和与之对应的 load 操作， 这条新指令的 执行需要花费 1 个时钟周期。加入这条指令后， branches 需要花费 5 个时钟周期。请计算新版本的 CPI 。

CPI=(35%\*35%\*1+35%\*(1-35%)\*1+(25%-35%\*35%)\*2+15%\*2+25%\*5)/(1-35%\*35%)=2.46

* 1. 如果新版本的时钟周期时间是旧版本的 1.2 倍， 那么哪个版本的 CPU 执行时间更短？

第一个：CPUtime=1.9\*1=1.9

第二个：CPUtime=2.46\*1.2=2.95

所以第一个短