内存系统，性能评测

于策

Outline

- 内存系统对性能的影响
- 性能评测
  - 基本性能指标
  - 加速比定律

Outline

- **内存系统对性能的影响**
- 性能评测
  - 基本性能指标
  - 加速比定律

内存系统对性能的影响

- 对于很多应用而言，瓶颈在于内存系统，而不是CPU
- 内存系统的性能包括两个方面：延迟和带宽
  - 延迟：处理器向内存发起访问直至获取数据所需要的时间
  - 带宽：内存系统向处理器传输数据的速率

延迟和带宽的区别

- 理解延迟与带宽的区别非常重要。
- 考虑消防龙头的情形。如果打开消防龙头后2秒水才从消防水管的尽头流出，那么这个系统的延迟就是2秒。
- 当水开始流出后，如果水管1秒钟能流出5加仑的水，那么这个水管的“带宽”就是5加仑/秒。
- 如果想立刻扑灭火灾，那么更重要是减少延迟的时间。
- 如果是希望扑灭更大的火，那么需要更高的带宽。

内存延迟示例

- 考虑某一处理器以1GHz（1纳秒时钟）运行，与之相连的DRAM有100纳秒的延迟（没有高速缓存）。假设处理器有两个multiply-add部件，在每1纳秒的周期内能执行4条指令。
  - 处理器的峰值是4GFLOPS。
  - 由于内存延迟是100个周期，并且块大小为一个字（word），每次处理内存访问请求时，处理器必须要等待100个周期，才能够获得数据。

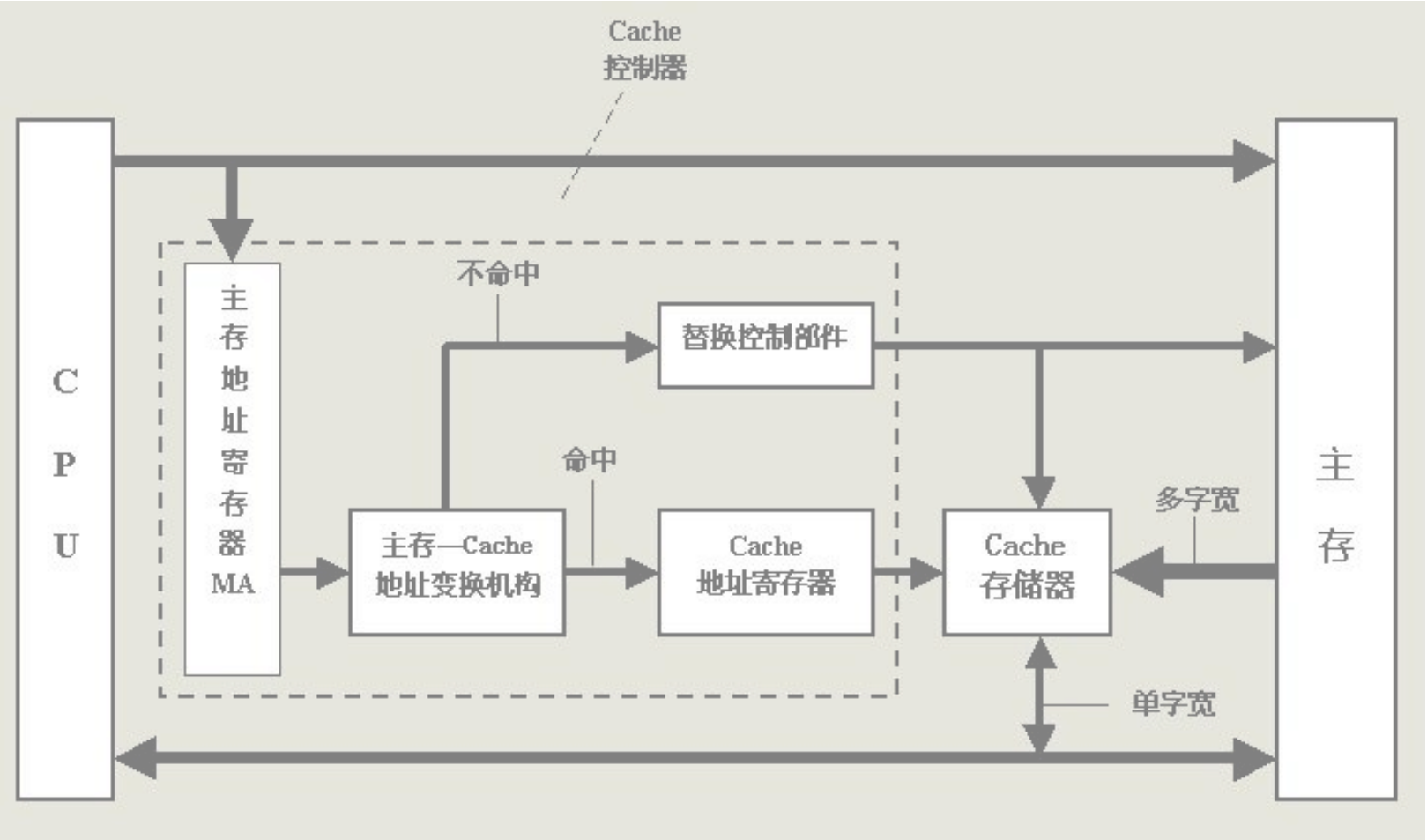
内存延迟示例

- 在以上平台上，考虑计算两个向量点积的问题。
  - 计算点积对每对向量元素进行一次乘法-加法运算，即每一次浮点运算需要取一次数据。
  - 此计算的峰值速度的限制是，每100纳秒才能够进行一次浮点计算，速度为10MFLOPS，只是处理器峰值速度的很小一部分。

使用高速缓存改善延迟

- 高速缓存是处理器与DRAM之间的更小但更快的内存单元。
- 这种内存是低延迟高带宽的存储器。
- 如果某块数据被重复使用，高速缓存就能减少内存系统的有效延迟
- 由高速缓存提供的数据份额称为高速缓存*命中率*(*hit ratio*)
- 高速缓存命中率严重影响内存受限程序的性能。

高速缓存


Cache
控制器
C
P
U
主
存
地
址
寄
存
器
MA
不命中
替换控制部件
命中
主存—Cache
地址变换机构
Cache
地址寄存器
Cache
存储器
多字宽
主
存
单字宽

缓存效果示例

- 继续考虑前一示例。
- 在其中加入一个大小为32KB，延迟时间为1纳秒(或1个周期)的高速缓存。
- 使用此系统来计算矩阵乘法，两个矩阵A和B的维数为32 × 32。
  - 之所以选择这个大小，是为了能够将A、B两个矩阵以及结果矩阵都放入高速缓存中。

缓存效果示例

- 结果如下
  - 将两个矩阵取到高速缓存中等同于取2K个字，需要大约200 μs。
  - 两个$n \times n$ 的矩阵乘需要$2n^3$ 步计算。在本例中，需要64K步计算，如果每个周期执行4条指令，则需要16K个周期，即 16 μs。
  - 总计算时间大约是加载存储时间以及计算时间之和，即200 + 16 μs。
  - 峰值计算速度为64K/216 =303 MFLOPS。

缓存的效果

- 对相同数据项的重复引用相当于“时间本地性(temporal locality)”
- 对于高速缓存的性能来说，数据的重复使用至关重要。

内存带宽的影响

- 内存带宽由内存总线的带宽和内存部件决定。
  - 可以通过增加内存块的大小来提高带宽。
- 底层系统在 L时间单位内(L为系统的延迟)存取B单位的数据(B为块大小)

内存带宽的影响示例

- 继续上一示例，将块大小由1个字改为4个字。同样考虑点积计算：
  - 假定向量数据在内存中线性排列，则在200个周期内能够执行8FLOPs(4次乘法-加法)
  - 这是因为每一次内存访问取出向量中4个连续的字
  - 因此，两次连续访问能够取出每个向量中的4个元素。
  - 这就相当于每25ns执行一次FLOP，即峰值速度为40MFLOPS。

内存带宽的影响

- 需要注意的是，增加块的大小，并不能改变系统的延迟。
- 物理上讲，本例中的情形可以认为是与多个存储区相连接的宽的数据总线(4个字，或者128位)
- 实际上，构建这样的宽总线的代价是昂贵的。
- 在更切实可行的系统中，得到第一个字后，连续的字在紧接着的总线周期里被送到内存总线。

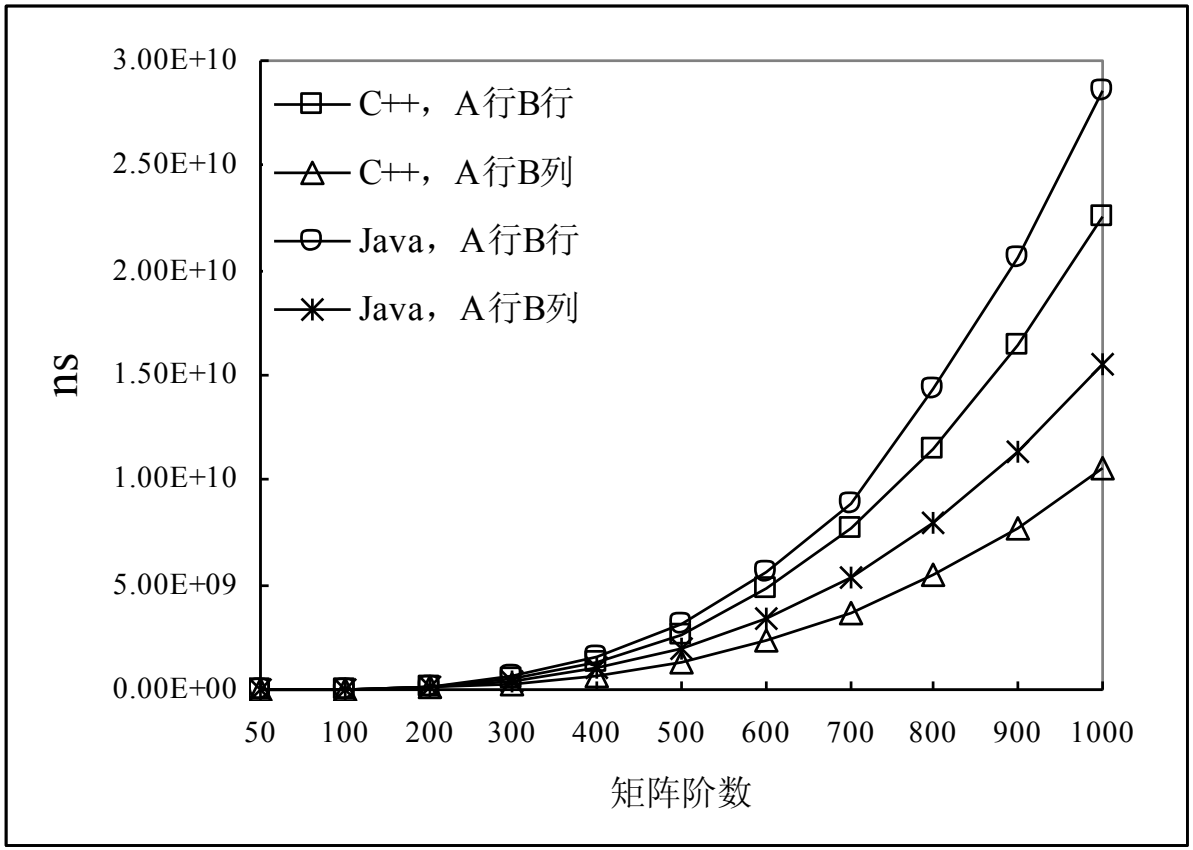
内存带宽的影响示例

- 以上示例清楚地说明了增加带宽对于提高峰值计算速度的影响。
- 对数据布局的假设是，连续的数据字被连续的指令所使用(空间本地性，spatial locality)
- 如果以数据布局为中心，那么计算的步骤应该确保连接的计算使用连接的数据

小结

- 以上示例阐述了如下概念：
  - 利用应用程序的空间本地性与时间本地性对于减少内存延迟及提高有效内存带宽非常重要。
  - 计算次数与内存访问次数的比是一个很好的预测内存带宽的承受程序的指标。
  - 内存的布局以及合理组织计算次序能对空间本地性和时间本地性产生重大影响。

实际测试



2.93GHz Intel 处理器，1M高速缓存，512M主存（533MHz）

Outline

- 内存系统对性能的影响
- **性能评测**
  - **基本性能指标**
  - 加速比定律

并行计算性能评测：CPU性能指标

- 工作负载
  - 执行时间 （Elapsed Time）
  - 浮点运算数（各种操作折算为浮点运算数的经验规则） ，Flop
  - 指令数目 ，MIPS
- 并行执行时间

T_{comput} 为计算时间，T_{paro} 为并行开销时间，T_{comm}为相互通信时间

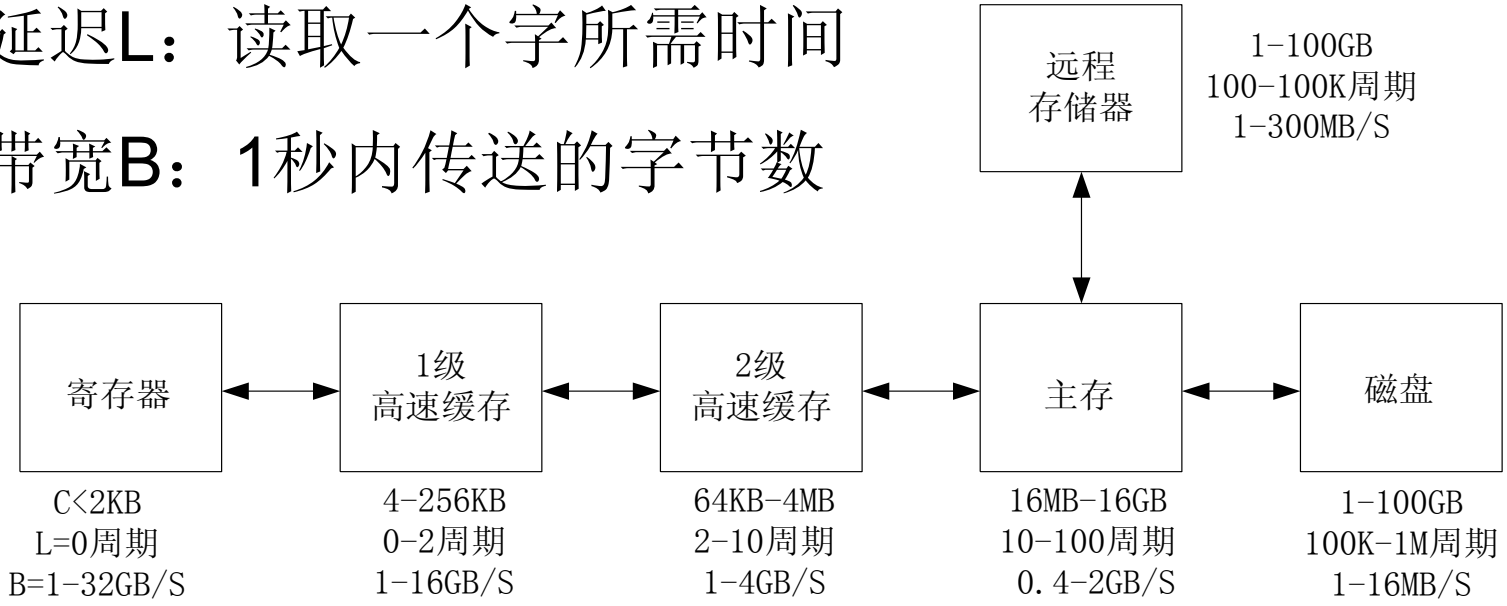
$$T_n = T_{comput} + T_{paro} + T_{comm}$$

并行计算性能评测：存储器层次

容量C：能保存数据的字节数

延迟L：读取一个字所需时间

带宽B：1秒内传送的字节数



并行计算性能评测：性能指标

- 计算/通信比

$$\frac{Computation\ Time}{Communication\ Time} = \frac{t_{comp}}{t_{comm}}$$

- 加速比

$$S(n) = \frac{Execution\ time\ (one\ processor\ system)}{Execution\ time\ (multiprocessor\ system)} = \frac{t_s}{t_p}$$

并行计算性能评测：性能指标

- 开销
  - 部分处理器的空闲
  - 并行版本中所需的顺序计算中不会出现的额外计算
  - 发送消息所需的通信时间
- 效率

$$E = \frac{\text{Execution time using one processor}}{\text{Execution time using a multiprocessor} \times \text{number of processors}}$$

$$= \frac{t_s}{t_p \times n}$$

并行计算性能评测：性能指标

- 代价
  - 代价＝（执行时间）×（所使用的处理器总数）

The *processor-time* product or *cost* (or *work*) of a computation defined as

$$\text{Cost} = (\text{execution time}) \times (\text{total number of processors used})$$

The cost of a sequential computation is simply its execution time, t_s. The cost of a parallel computation is $t_p \times n$. The parallel execution time, t_p, is given by $t_s/S(n)$. Hence, the cost of a parallel computation is given by

$$\text{Cost} = \frac{t_s n}{S(n)} = \frac{t_s}{E}$$

参数定义

- P：处理器数；
- W：问题规模（计算负载、工作负载，给定问题的总计算量）；
  - W_s：应用程序中的串行分量，f是串行分量比例（$f = W_s/W$）；
  - W_P：应用程序中可并行化部分，1-f为并行分量比例；
  - $W_s + W_p = W$；
- T_s：串行执行时间，T_p：并行执行时间；
- S：加速比，E：效率。

Outline

- 内存系统对性能的影响
- **性能评测**
  - 基本性能指标
  - **加速比定律**

加速比性能定律

- Amdahl 定律
- Gustafson 定律
- Sun and Ni 定律

加速比性能定律

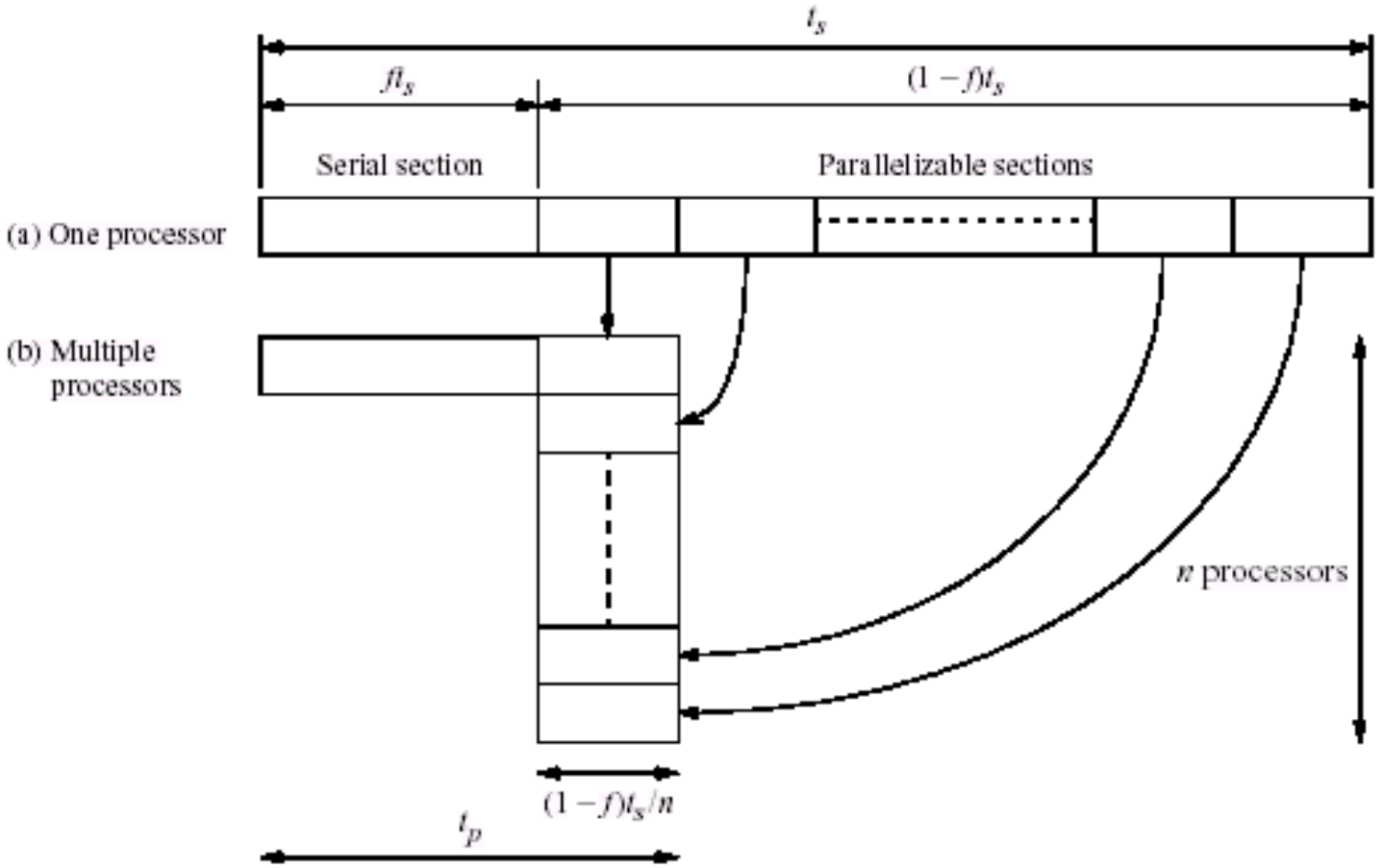
- **Amdahl 定律**
- Gustafson 定律
- Sun and Ni 定律

Amdahl 定律

1922-

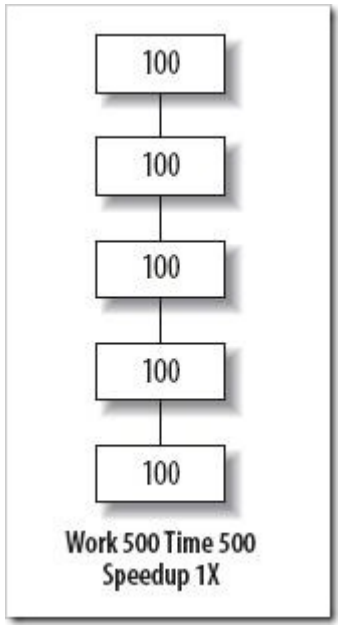
- 出发点：
  - 固定不变的计算负载；
  - 固定的计算负载分布在多个处理器上的，
  - 增加处理器加快执行速度，从而达到加速的目的。

Amdahl 定律：应用情形



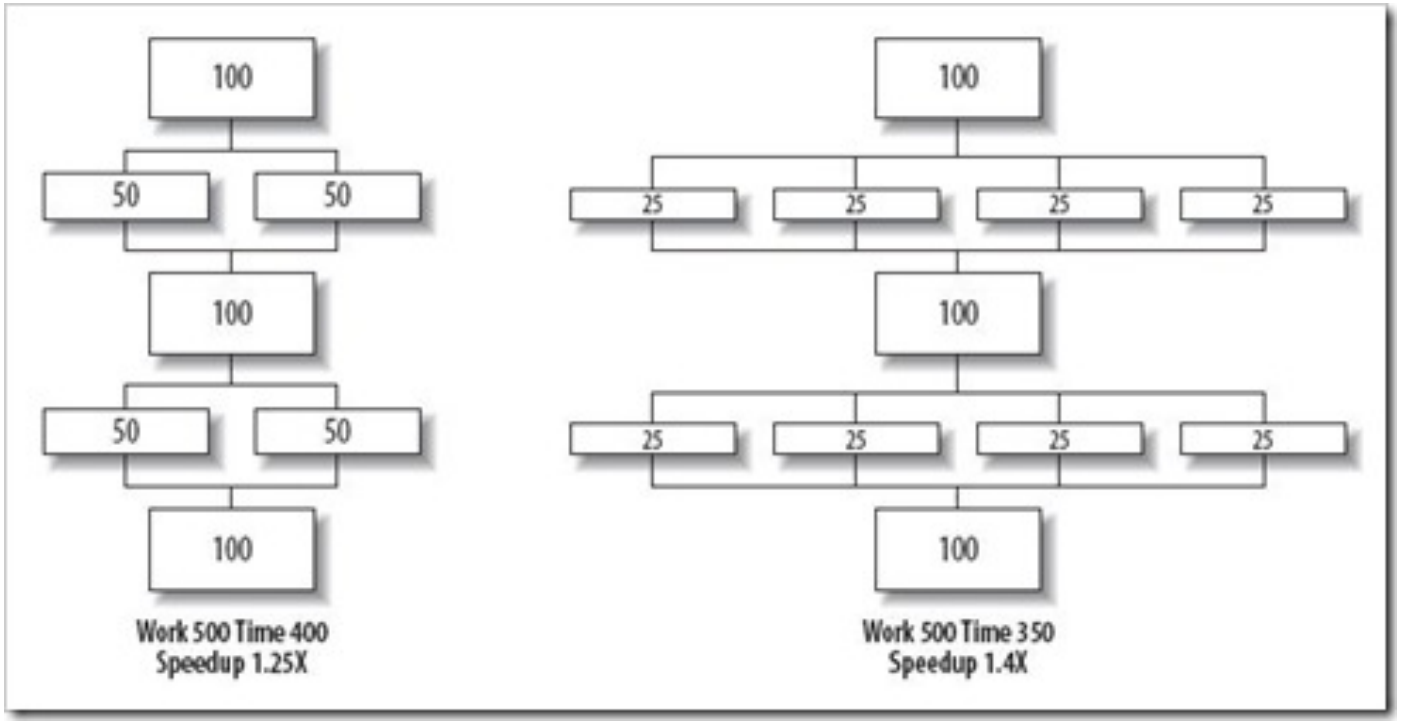
Amdahl 定律：示例（1）

- 程序由 5 个相同的部分组成，且每个部分运行时间均花费 100 秒



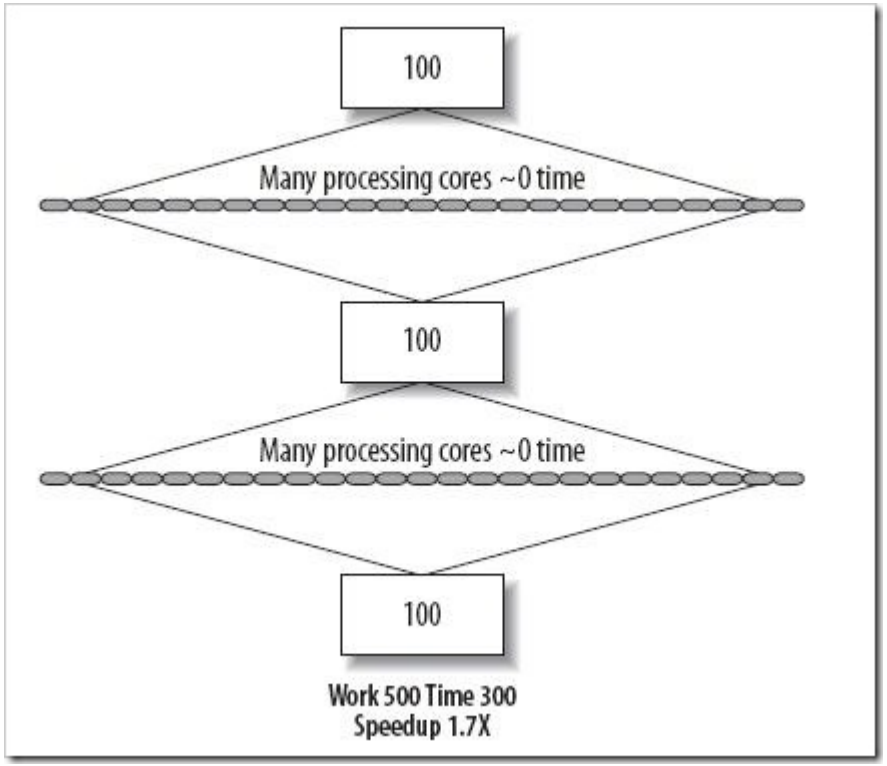
Amdahl 定律：示例（2）

- 并行化其中两个部分



Amdahl 定律：示例（3）

- 增加处理器数



Amdahl 定律：公式

- 固定负载的加速公式：

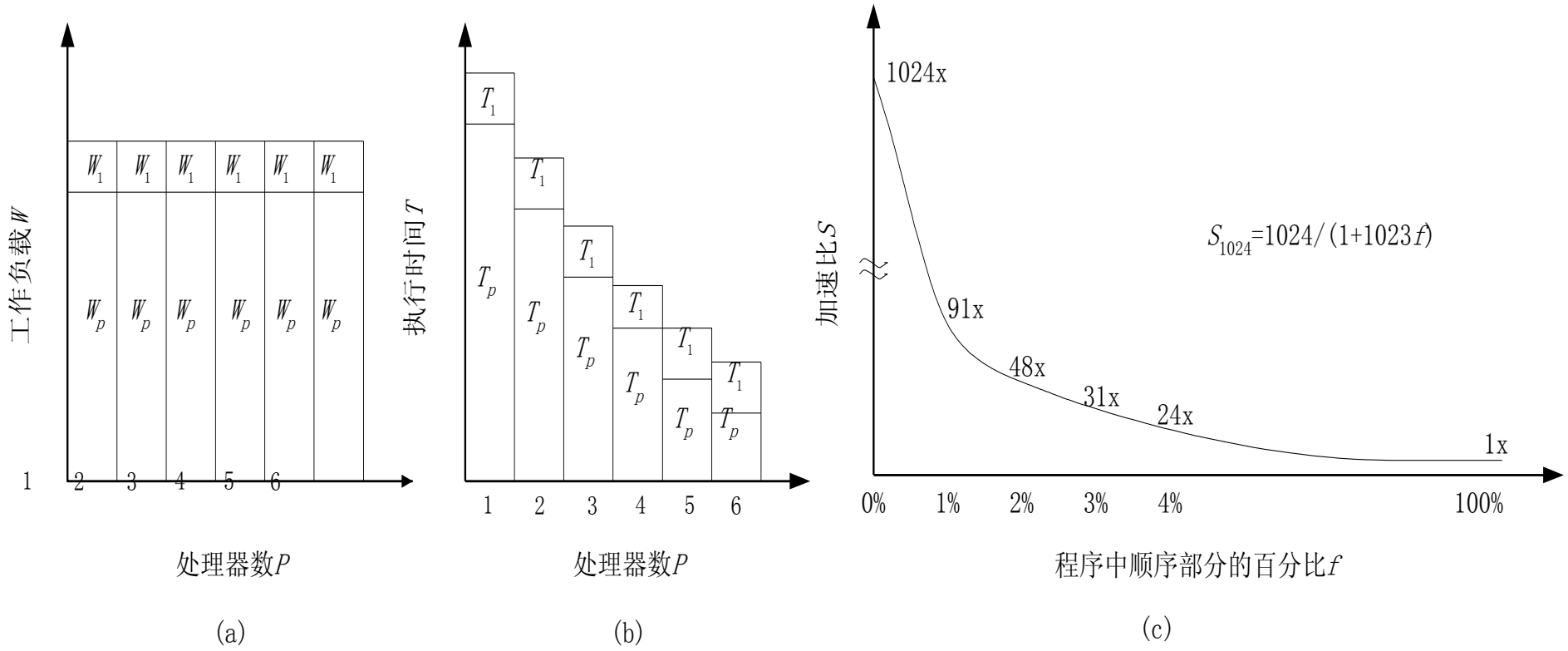
$$S = \frac{Ws + Wp}{Ws + W_P / p}$$

- $W_s + W_p$可相应地表示为f+（1-f）

$$S = \frac{f + (1-f)}{f + \frac{1-f}{p}} = \frac{p}{1 + f(p-1)}$$

- p→∞时，上式极限为：　　S= 1 / f

Amdahl 定律：曲线图



Amdahl 定律：考虑额外开销

- W_o为额外开销

$$S = \frac{W_S + W_P}{W_S + \frac{W_P}{p} + W_O} = \frac{W}{fW + \frac{W(1-f)}{p} + W_O} = \frac{p}{1 + f(p-1) + W_O p / W}$$

加速比性能定律

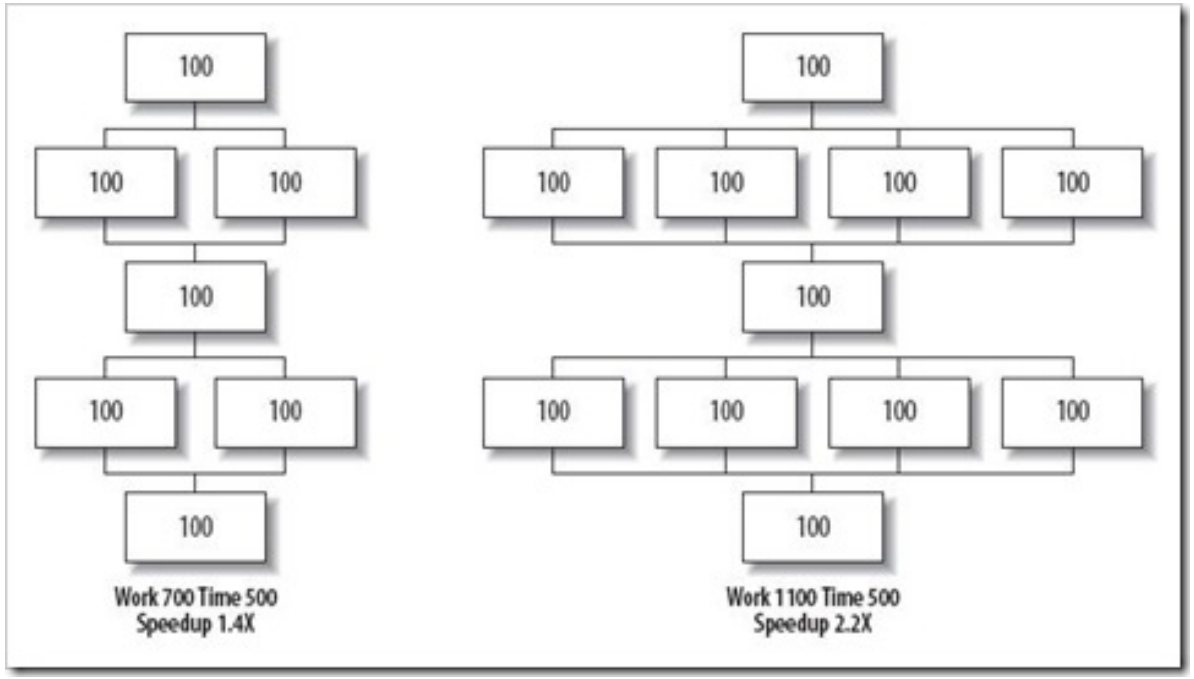
- Amdahl 定律
- **Gustafson 定律**
- Sun and Ni 定律

Gustafson 定律

1955-

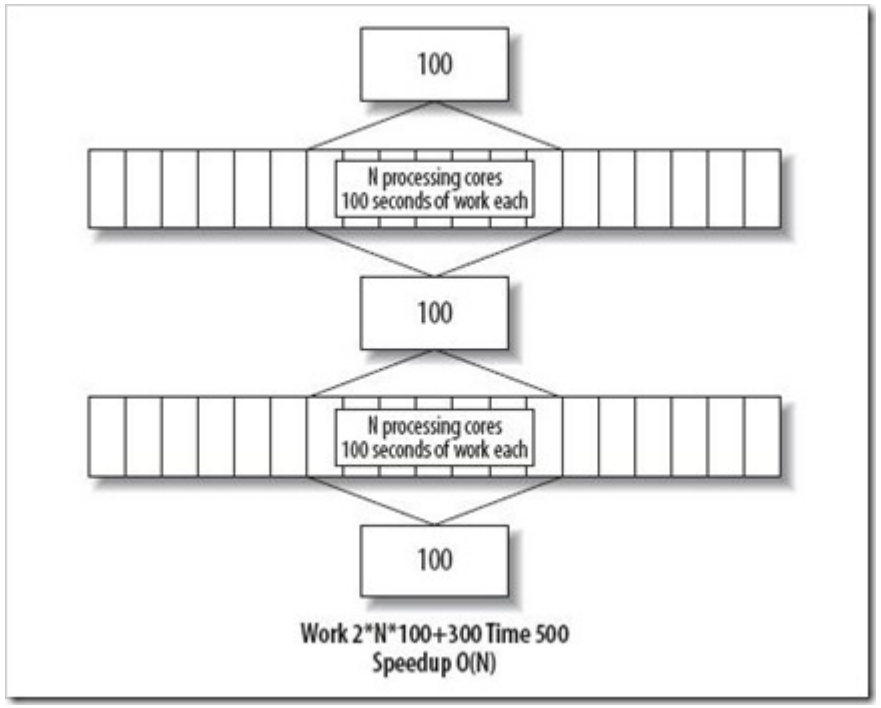
- 出发点：
  - 对于很多大型计算，精度要求很高，即在此类应用中精度是个关键因素，而计算时间是固定不变的。此时为了提高精度，必须加大计算量，相应地亦必须增多处理器数才能维持时间不变；
  - 除非学术研究，在实际应用中没有必要固定工作负载而计算程序运行在不同数目的处理器上，增多处理器必须相应地增大问题规模才有实际意义。

并行的同时增加计算量


100
100
100
100
100
100
100
Work 700 Time 500
Speedup 1.4X
100
100
100
100
100
100
100
100
100
Work 1100 Time 500
Speedup 2.2X

增加处理器数与计算量

- 串行部分仍然花费相同的时间量，随着其在整体中所占比例的下降，变得越来越不重要。



Gustafson 定律：公式

- Gustafson加速定律：

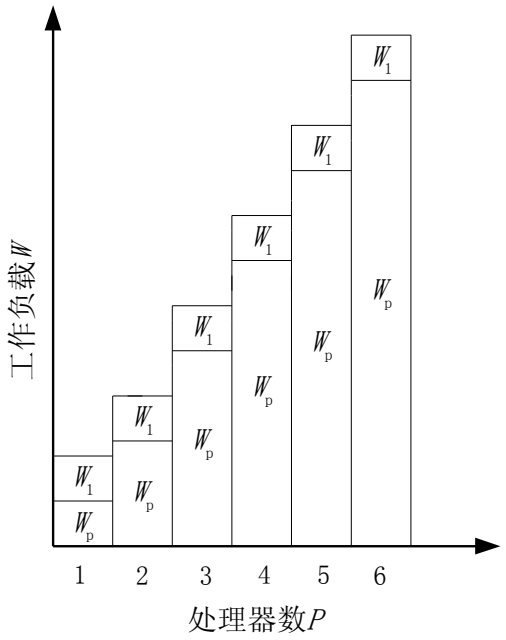
$$S' = \frac{W_S + pWp}{W_S + p \cdot Wp / p} = \frac{W_S + pWp}{W_S + W_P}$$

$$S' = f + p(1-f) = p + f(1-p) = p - f(p-1)$$

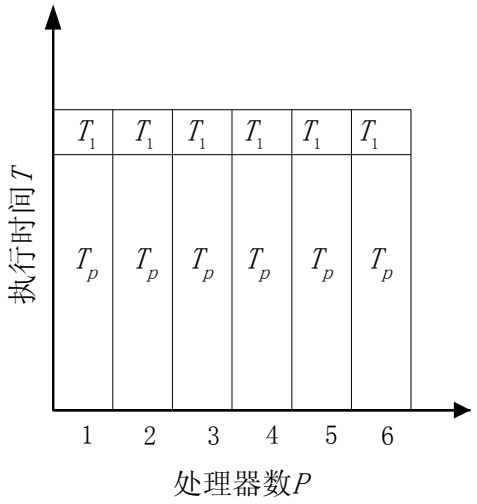
- 并行开销W_o：

$$S' = \frac{W_S + pW_P}{W_S + W_P + W_O} = \frac{f + p(1-f)}{1 + W_O / W}$$

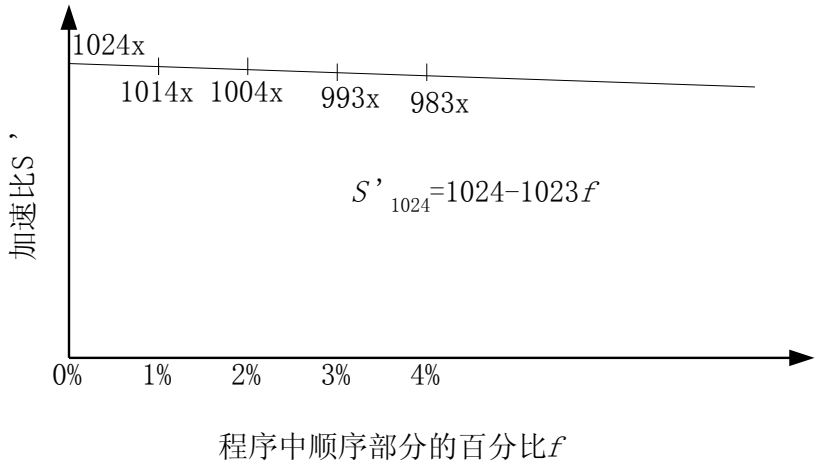
Gustafson 定律：曲线图



(a)



(b)



(c)

加速比性能定律

- Amdahl 定律
- Gustafson 定律
- **Sun and Ni 定律**

Sun-Ni定律

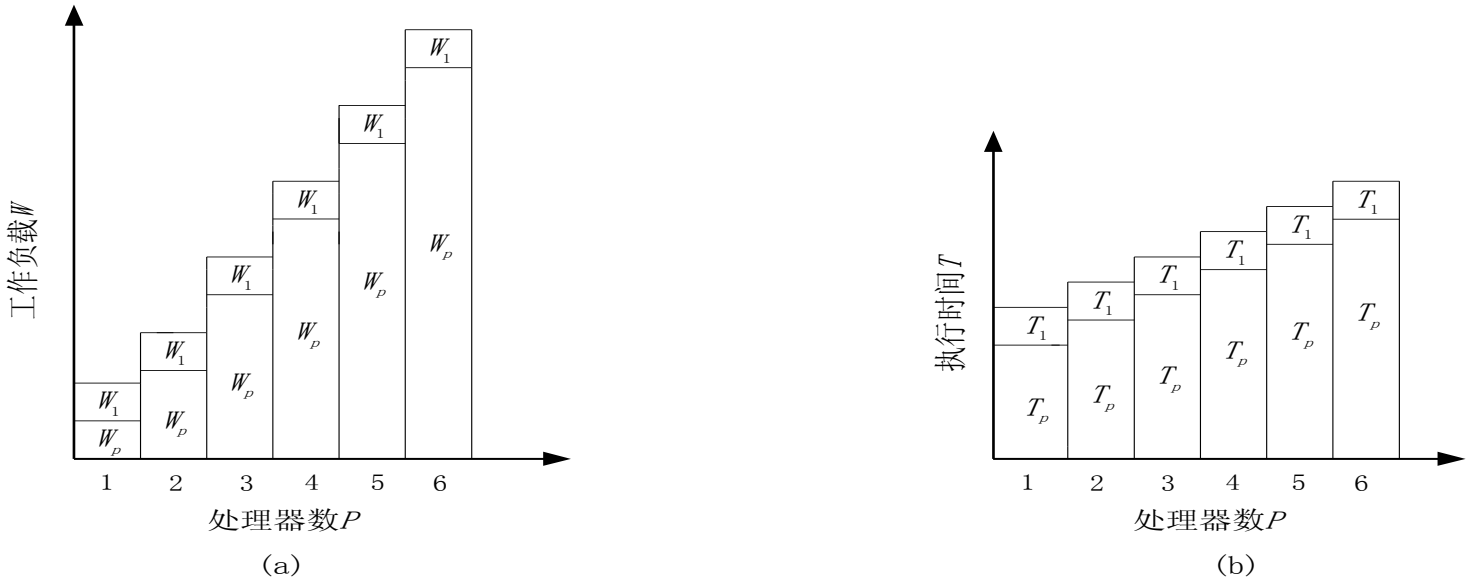
- 基本思想：
  - 只要存储空间许可，应尽量增大问题规模以产生更好和更精确的解（此时可能使执行时间略有增加）。
  - 假定在单节点上使用了全部存储容量M并在相应于W的时间内求解之，此时工作负载W= fW + （1-f）W。
  - 在p 个节点的并行系统上，能够求解较大规模的问题是因为存储容量可增加到pM。令因子G（p）反应存储容量增加到p倍时并行工作负载的增加量，所以扩大后的工作负载W = fW + （1-f）G（p）W。
- 存储受限的加速公式 ：

$$S'' = \frac{fW + (1-f)G(p)W}{fW + (1-f)G(p)W/p} = \frac{f + (1-f)G(p)}{f + (1-f)G(p)/p}$$

- 并行开销W_O:

$$S' = \frac{fW + (1-f)WG(p)}{fW + (1-f)G(p)W/p + W_O} = \frac{f + (1-f)G(p)}{f + (1-f)G(p)/p + W_O/W}$$

Sun-Ni定律(cont’d)



- G（p）=1时就是Amdahl加速定律；
- G（p）=p 变为 f + p（1-f），就是Gustafson加速定律
- G（p）>p时，相应于计算机负载比存储要求增加得快，此时 Sun和 N i 加速均比 Amdahl 加速和 Gustafson 加速为高。