第4章 向量处理机

- 向量由一组有序、具有相同类型和位数的元素组成。
- 在流水线处理机中，设置向量数据表示和相应的向量指令，称为向量处理机。
- 不具有向量数据表示和相应的向量指令的流水线处理机，称为标量处理机。
- 典型的向量处理机
  - 1976年 Cray-1超级计算机

    浮点运算速度达到了每秒1亿次
  - CDC Cyber 205，Cray Y-MP，NEC SX-X/44，Fujitsu VP2600等

    性能达到了每秒几十亿～几百亿次浮点运算

4.1 向量的处理方式

以计算表达式 D=A×（B－C）为例

A、B、C、D —— 长度为 N 的向量

```
main()
{// 设A、B、C、D都是长度为n的数组
    for(int i=0;i<n;i++)
        D[i]=A[i]*(B[i]-C[i]);
}
```

1. 横向(水平)处理方式

- 向量计算是按行的方式从左到右横向地进行。
  - 先计算：　$d_1 \leftarrow a_1 \times (b_1 - c_1)$
  - 再计算：　$d_2 \leftarrow a_2 \times (b_2 - c_2)$
  - ……
  - 最后计算：　$d_N \leftarrow a_N \times (b_N - c_N)$
- 组成循环程序进行处理。

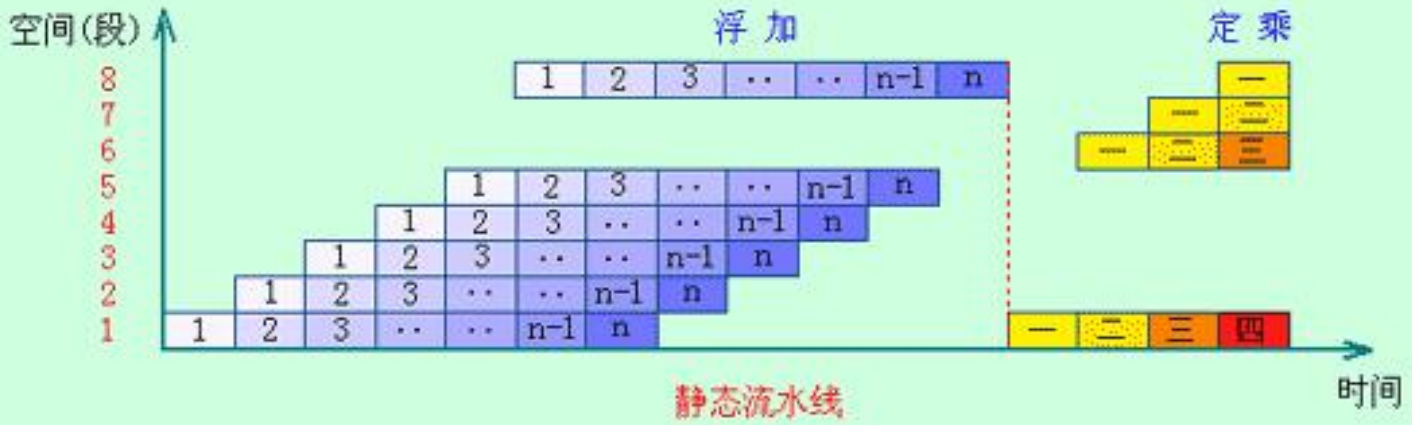
$$q_i \leftarrow b_i - c_i$$

$$d_i \leftarrow q_i \times a_i$$

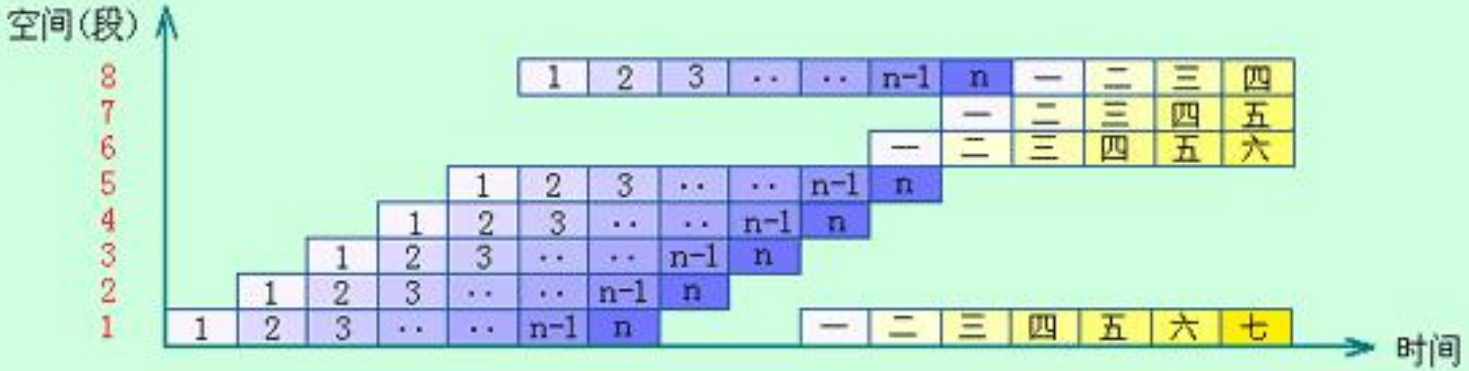
  - 数据相关：N次　　功能切换：2N次
- 不适合于向量处理机的并行处理。

静、动态流水线的时空图

假设该流水线要先做几个浮点加法，然后再做一批定点乘法。



静态流水线



动态流水线

2. 纵向 (垂直)处理方式

- 向量计算是按列的方式从上到下纵向地进行。

先计算
$$\begin{cases} q_1 \leftarrow b_1 - c_1 \\ \cdots\cdots \\ q_N \leftarrow b_N - c_N \end{cases}$$

再计算
$$\begin{cases} d_1 \leftarrow q_1 \times a_1 \\ \cdots\cdots \\ d_N \leftarrow q_N \times a_N \end{cases}$$

- 表示成向量指令：

$$Q = B - C$$

$$D = Q \times A$$

  - 两条向量指令之间：

    功能切换：1次

3. 纵横 (分组)处理方式

- 又称为分组处理方式。
- 把向量分成若干组，组内按纵向方式处理，依次处理各组。
- 对于上述的例子，设：

$$N=S\times n+r$$

  - 其中N为向量长度，S为组数，n为每组的长度，r为余数。
  - 若余下的r个数也作为一组处理，则共有S+1组。

- 运算过程为：

- 先算第1组：

  $Q_{1\sim n} \leftarrow B_{1\sim n} - C_{1\sim n}$

  $D_{1\sim n} \leftarrow Q_{1\sim n} \times A_{1\sim n}$

- 再算第2组：

  $Q_{(n+1)\sim 2n} \leftarrow B_{(n+1)\sim 2n} - C_{(n+1)\sim 2n}$

  $D_{(n+1)\sim 2n} \leftarrow Q_{(n+1)\sim 2n} \times A_{(n+1)\sim 2n}$

- 依次进行下去，直到最后一组：第S+1组。
- 每组内各用两条向量指令。

4.2 向量处理机的结构

- 向量处理机的结构因具体机器不同而不同。

  由所采用的向量处理方式决定。

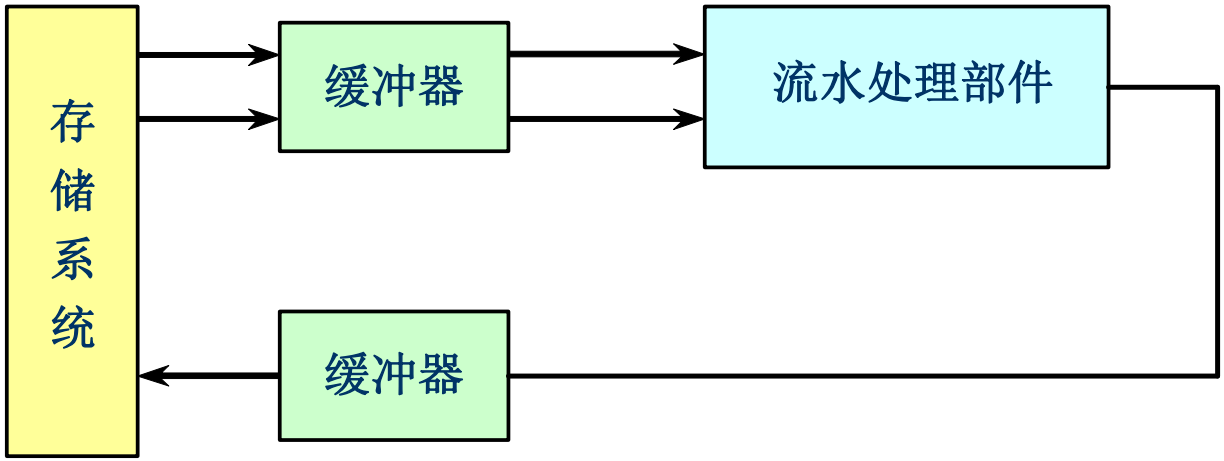
- 有两种典型的结构
  - 存储器-存储器型结构

    纵向处理方式采用
  - 寄存器-寄存器型结构

    分组处理方式采用

4.2.1 “存储器-存储器”结构

1. 采用纵向处理方式的向量处理机对处理机结构的要求：存储器－存储器结构
   - 向量指令的源向量和目的向量都是存放在存储器中，运算的中间结果需要送回存储器。
   - 流水线运算部件的输入和输出端都直接（或经过缓冲器）与存储器相联，从而构成存储器-存储器型操作的运算流水线。
     - 例如：STAR-100、CYBER-205



“存储器—存储器”型操作的运算流水线

2. 要充分发挥这种结构的流水线效率，存储器要不断地提供源操作数，并不断地从运算部件接收结果。

（每拍从存储器读取两个数据，并向存储器写回一个结果）

- 对存储器的带宽以及存储器与处理部件的通信带宽提出了非常高的要求。
- 解决方法：一般是通过采用多体交叉并行存储器和缓冲器技术。

例如，70年代初问世的Star 100

- 存储器：32个体交叉
- 每个体的数据宽度：8个字（字长64位）
- 最大数据流量：每秒2亿字

4.2.2 “寄存器-寄存器”结构

在向量的分组处理方式中，对向量长度N没有限制，但组的长度n却是固定不变的。

- 对处理机结构的要求：寄存器－寄存器结构
- 设置能快速访问的向量寄存器，用于存放源向量、目的向量及中间结果。让运算部件的输入、输出端都与向量寄存器相联，就构成了“寄存器－寄存器”型操作的运算流水线。
  - 典型的寄存器－寄存器结构的向量处理机

    美国的CRAY-1、我国的YH-1巨型机

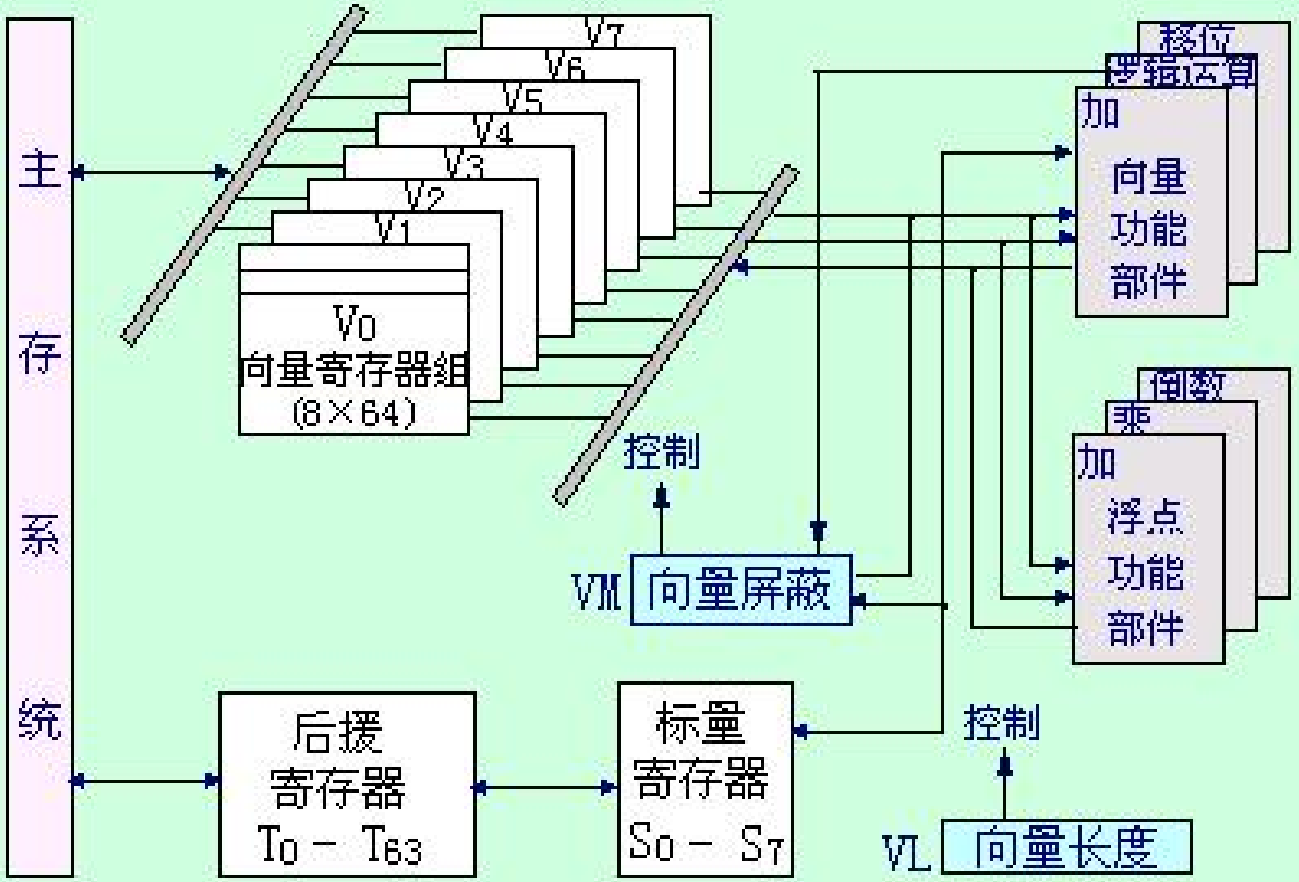
- 以CRAY-1机为例
  - 美国CRAY公司
  - 1976年
  - 每秒1亿次浮点运算
  - 时钟周期：12.5ns

1. CRAY-1的基本结构

- 功能部件

共有12条可并行工作的单功能流水线，可分别流水地进行地址、向量、标量的各种运算。

CRAY-1的基本结构


主
存
系
统
V7
V6
V5
V4
V3
V2
V1
V0
向量寄存器组
(8×64)
移位
逻辑运算
加
向量
功能
部件
倒数
乘
加
浮点
功能
部件
控制
VM
向量屏蔽
后援
寄存器
T0 - T63
标量
寄存器
S0 - S7
控制
VL
向量长度

- 6个单功能流水部件：进行向量运算
  - 整数加（3拍）
  - 逻辑运算（2拍）
  - 移位（4拍）
  - 浮点加（6拍）
  - 浮点乘（7拍）
  - 浮点迭代求倒数（14拍）

括号中的数字为其流水经过的时间，每拍为一个时钟周期，即12.5ns。

- 向量寄存组V
  - 由512个64位的寄存器组成，分成8组。
  - 组编号：V_0～V_7
  - 每一个组称为一个向量寄存器，可存放一个长度（即元素个数）不超过64的向量。
  - 每个向量寄存器可以每拍向功能部件提供一个数据元素，或者每拍接收一个从功能部件来的结果元素。
- 标量寄存器S和快速暂存器T
  - 标量寄存器有8个：S_0～S_7　64位
  - 快速暂存器T用于在标量寄存器和存储器之间提供缓冲。

- 向量屏蔽寄存器VM
  - 64位，每一位对应于向量寄存器的一个单元。
  - 作用：用于向量的归并、压缩、还原和测试操作、对向量某些元素的单独运算等。

1. CRAY-1向量处理的一个显著特点
   - 每个向量寄存器V_i都有连到6个向量功能部件的单独总线。
   - 每个向量功能部件也都有把运算结果送回向量寄存器组的总线。

- 只要不出现V_i冲突和功能部件冲突，各V_i之间和各功能部件之间都能并行工作，大大加快了向量指令的处理。
  - V_i冲突：并行工作的各向量指令的源向量或结果向量使用了相同的V_i。

  例如：源向量相同

  $$V_3 \leftarrow V_1 + V_2$$

  $$V_5 \leftarrow V_4 \wedge V_1$$

  - 功能部件冲突：并行工作的各向量指令要使用同一个功能部件。

4.3 提高向量处理机性能的常用技术

提高向量处理机性能的方法

- 设置多个功能部件，使它们并行工作；
- 采用链接技术，加快一串向量指令的执行；
- 采用分段开采技术，加快循环处理；
- 采用多处理机系统，进一步提高性能。

4.3.1 设置多个功能部件

- 设置多个独立的功能部件。这些部件能并行工作，并各自按流水方式工作，从而形成了多条并行工作的运算操作流水线。

  例如：CRAY-1向量处理机有4组12个单功能流水部件：

  - 向量部件：向量加，移位，逻辑运算
  - 浮点部件：浮点加，浮点乘，浮点求倒数
  - 标量部件：标量加，移位，逻辑运算，数“1”/计数
  - 地址运算部件：整数加，整数乘

4.3.2 链接技术

1. 两条向量指令占用功能流水线和向量寄存器的4种情况

- 指令不相关

  例如：V0←V1＋V2

  V6←V4*V5

  - 这两条指令分别使用各自所需的流水线和向量寄存器，可以并行执行。

- 功能部件冲突

  例如：V3←V1＋V2

  V6←V4＋V5

- 这两条指令都要使用加法流水线，发生了功能部件冲突（但向量寄存器不冲突）。当第一条指令流出时，占用加法流水线。第二条指令要等加法流水线变成空闲后，才能流出。

➢ 源寄存器冲突

例如：V3←V1＋V2

V6←V1*V4

- 这两条向量指令的源向量之一都取自V1。由于两者的首元素下标可能不同，向量长度也可能不同，所以难以由V1同时提供两条指令所需要的源向量。
- 这两条向量指令不能同时执行。只有等第一条向量指令执行完、释放V1之后，第二条向量指令才能开始执行。

- 结果寄存器冲突

  两条向量指令使用了相同的结果向量寄存器。

  例如：V4←V1＋V2

  V4←V3*V5

  - 这两条指令都要访问目的寄存器V4。由于第一条指令在先，所以它先占用V4直到运算完成，然后再流出后一条指令。

2. 当前一条指令的结果寄存器是后一条指令的源寄存器、且不存在任何其他冲突时，就可以用链接技术来提高性能。

   例如：V3←V1＋V2

   V6←V3*V4

- 向量流水线链接：具有先写后读相关的两条指令，在不出现功能部件冲突和源向量冲突的情况下，可以把功能部件链接起来进行流水处理，以达到加快执行的目的。
  - Cray-1向量处理的一个显著特点
  - 链接特性的实质

    把流水线定向的思想引入到向量执行过程的结果。

- 由于同步的需求，链接时，Cray-1中把向量数据元素送往向量功能部件以及把结果存入向量寄存器都需要一拍时间，从存储器中把数据送入访存功能部件也需要一拍时间。

➢ 进行向量链接的要求

保证：无向量寄存器使用冲突和无功能部件使用冲突

- 只有在前一条指令的第一个结果元素送入结果向量寄存器的那一个时钟周期才可以进行链接。
- 当一条向量指令的两个源操作数分别是两条先行指令的结果寄存器时，要求先行的两条指令产生运算结果的时间必须相等，即要求有关功能部件的通过时间相等。
- 要进行链接执行的向量指令的向量长度必须相等，否则无法进行链接。

4.3.3 分段开采技术

如果向量的长度大于向量寄存器的长度，

该如何处理呢？

- 当向量的长度大于向量寄存器的长度时，必须把长向量分成长度固定的段，然后循环分段处理，每一次循环只处理一个向量段。
- 这种技术称为分段开采技术。
  - 由系统硬件和软件控制完成，对程序员是透明的。

4.3.4 采用多处理机系统

许多新型向量处理机系统采用了多处理机系统结构。例如：

- CRAY-2
  - 包含了4个向量处理机
  - 浮点运算速度最高可达1800MFLOPS
- CRAY Y-MP、C90

  最多可包含16个向量处理机

4.4 向量处理机的性能评价

衡量向量处理机性能的主要参数：

- 向量指令的处理时间
- 向量长度为无穷大时的向量处理机的最大性能
- 半性能向量长度
- 向量长度临界值

4.4.1 向量指令的处理时间T_{vp}

1. 一条向量指令的处理时间T_{vp}

- 执行一条向量长度为n的向量指令所需的时间为：

$$T_{vp} = T_s + T_e + (n-1)T_c$$

- T_s：向量处理部件流水线的建立时间

  为了使处理部件流水线能开始工作（即开始流入数据）所需要的准备时间。

- T_e：向量流水线的通过时间

第一对向量元素通过流水线并产生第一个结果所花的时间 。

- T_c：流水线的时钟周期时间

- 把上式中的参数都折算成时钟周期个数：

$$T_{vp} = [s + e + (n-1)]T_c$$

  - s：T_s所对应的时钟周期数
  - e：T_e所对应的时钟周期数

- 不考虑T_s，并令T_{start}=e－1

$$T_{vp} = (T_{start} + n)T_c$$

  - T_{start}：从一条向量指令开始执行到还差一个时钟周期就产生第一个结果所需的时钟周期数。可称之为该向量指令的启动时间。此后，便是每个时钟周期流出一个结果，共有n个结果。

1. 一组向量指令的处理时间

- 对于一组向量指令而言，其执行时间主要取决于三个因素：
  - 向量的长度
  - 向量操作之间是否存在流水功能部件的使用冲突
  - 数据的相关性
- 把能在同一个时钟周期内一起开始执行的几条向量指令称为一个编队。

- 可以看出：同一个编队中的向量指令之间一定不存在流水向量功能部件的冲突和数据的冲突。

➢ 编队后，这个向量指令序列的总的执行时间为各编队的执行时间的和 。

$$T_{all} = \sum_{i=1}^{m} T_{vp}^{(i)}$$

- $T^{(i)}_{vp}$：第i个编队的执行时间
- m：编队的个数

- 当一个编队是由若干条指令组成时，其执行时间就应该由该编队中各指令的执行时间的最大值来确定。

$T^{(i)}_{start}$：第i编队中各指令的启动时间的最大值

$$T_{all}=\sum_{i=1}^{m}T_{vp}^{(i)}=\sum_{i=1}^{m}(T_{start}^{(i)}+n)T_c=(\sum_{i=1}^{m}T_{start}^{(i)}+mn)T_c=(T_{start}+mn)T_c$$

$$T_{start}=\sum_{i=1}^{m}T_{start}^{(i)}$$

该组指令总的启动时间（时钟周期个数）

- 表示成时钟周期个数

$T_{all} = T_{start} + mn$ （拍）

分段开采时一组向量指令的总执行时间

- 当向量长度n大于向量寄存器长度MVL时，需要分段开采。
- 引入一些额外的处理操作

（假设：这些操作所引入的额外时间为T_{loop}个时钟周期）

  - 设 $\left\lfloor \frac{n}{MVL} \right\rfloor = p$　　q：余数
  - 共有m个编队

- 对于最后一次循环来说，所需要的时间为：

$$T_{last} = T_{start} + T_{loop} + m \times q$$

- 其他的每一次循环所要花费的时间为：

$$T_{step} = T_{start} + T_{loop} + m \times MVL$$

- 总的执行时间为：

$$\begin{aligned} T_{all} &= T_{step} \times p + T_{last} \\ &= (T_{start} + T_{loop} + m \times MVL) \times p + (T_{start} + T_{loop} + m \times q) \\ &= (p+1) \times (T_{start} + T_{loop}) + m(MVL \times p + q) \\ &= \left\lceil \frac{n}{MVL} \right\rceil \times (T_{start} + T_{loop}) + mn \end{aligned}$$

4.4.2 最大性能R_∞和半性能向量长度$n_{1/2}$

1. 向量处理机的峰值性能 R_∞

- R_∞表示当向量长度为无穷大时，向量处理机的最高性能，也称为**峰值性能**。

$$R_\infty = \lim_{n\to\infty} \frac{\text{向量指令序列中浮点运算次数} \times \text{时钟频率}}{\text{向量指令序列执行所需的时钟周期数}}$$

- 半性能向量长度$n_{1/2}$
  - 半性能向量长度$n_{1/2}$是指向量处理机的性能为其最大性能的一半时所需的向量长度。
  - 评价向量流水线的建立时间对性能影响的重要参数。

- 向量长度临界值n_v
  - 向量长度临界值n_v是指：对于某一计算任务而言，向量方式的处理速度优于标量串行方式处理速度时所需的最小向量长度。