



# [12] 发明专利申请公开说明书

[21] 申请号 200410030669.1

[43] 公开日 2005年10月5日

[11] 公开号 CN 1677946A

[22] 申请日 2004.4.2

[21] 申请号 200410030669.1

[71] 申请人 华为技术有限公司

地址 518129 广东省深圳市龙岗区坂田华为  
总部办公楼

[72] 发明人 程智辉 李伟东

[74] 专利代理机构 北京集佳知识产权代理有限公司

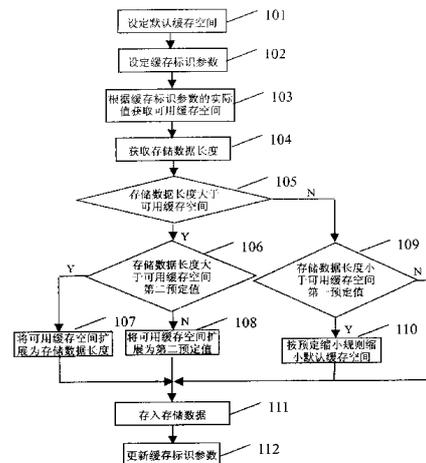
代理人 王学强

权利要求书 3 页 说明书 16 页 附图 4 页

[54] 发明名称 一种缓存分配方法及装置

[57] 摘要

本发明公开了一种缓存分配方法及装置，该方法包括：设定默认缓存空间；获取存储数据长度和可用缓存空间；当存储数据长度大于可用缓存空间时，按预定扩展规则扩展默认缓存空间；当存储数据长度小于可用缓存空间第一预定值时，按预定缩小规则缩小默认缓存空间。本发明提供的装置包括：参数维护装置、缓存空间获取装置、比较装置、缓存调整装置、缓存分配单元。利用本发明，可以有效地避免数据的溢出及缓存空间的浪费，同时减少缓存动态调整的频率，提高系统整体效率。



1、一种缓存分配方法，其特征在于，包括如下步骤：

A、设定默认缓存空间；

B、获取存储数据长度和可用缓存空间；

C、当所述存储数据长度大于所述可用缓存空间时，按预定扩展规则扩展所述默认缓存空间；

D、当所述存储数据长度小于所述可用缓存空间第一预定值时，按预定缩小规则缩小所述默认缓存空间。

2、如权利要求1所述的缓存分配方法，其特征在于，所述默认缓存空间的末端包括至少一个空位。

3、如权利要求1或2所述的缓存分配方法，其特征在于，所述步骤B包括：

B1、设定缓存标识参数；

B2、根据所述缓存标识参数的实际值获取所述可用缓存空间。

4、如权利要求3所述的缓存分配方法，其特征在于，所述缓存标识参数包括：已用缓存的起始指针、已用缓存的终止指针、缓存满标识、缓存空标识。

5、如权利要求4所述的缓存分配方法，其特征在于，所述步骤B1包括：

当所述默认缓存空间全部可用时，设定所述缓存满标识为假、缓存空标识为真；

当所述默认缓存空间部分可用时，设定所述缓存满标识为假、缓存空标识为假；

当所述默认缓存空间都不可用时，设定所述缓存满标识为真、缓存空标识为假。

6、如权利要求1或2所述的缓存分配方法，其特征在于，所述步骤C中的预定扩展规则包括：

C1、如果所述存储数据长度大于所述可用缓存空间第二预定值，则将所述可用缓存空间扩展为所述存储数据长度；

C2、如果所述存储数据长度小于所述可用缓存空间第二预定值，则将所述可用缓存空间扩展为第二预定值。

7、如权利要求 1 或 2 所述的缓存分配方法，其特征在于，所述步骤 D 包括：当所述存储数据长度小于所述可用缓存空间的一半时，将所述可用缓存空间缩小一半。

8、如权利要求 5 所述的缓存分配方法，其特征在于，所述步骤 C 和步骤 D 中存入所述存储数据时，

当所述缓存满标识及所述缓存空标识均为假时，判断所述已用缓存的终止指针是否到达所述默认缓存空间末端的空位前；

如果已到达，则从所述默认缓存空间的起始位置开始依次存入所述存储数据；

如果未到达，则从所述已用缓存的终止指针的下一个位置开始依次存入所述存储数据。

9、一种缓存分配装置，其特征在于，包括：

参数维护装置，用于维护缓存标识参数；

缓存空间获取装置，用于根据所述参数维护装置中的缓存标识参数获取可用缓存空间；

存储数据获取装置，用于获取存储数据及其长度；

缓存调整装置，用于根据所述缓存空间获取装置和所述存储数据获取装置的输出调整缓存空间的长度；

缓存分配单元，用于依次将所述存储数据存入经过所述缓存调整装置调整后的可用缓存空间。

10、如权利要求 9 所述的缓存分配装置，其特征在于，所述参数维护装

置包括:

参数设定装置, 用于设定所述缓存标识参数的初始值;

参数更新装置, 用于根据所述缓存调整装置的调整结果及所述缓存分配单元的分配结果更新所述缓存标识参数。

11、如权利要求 9 所述的缓存分配装置, 其特征在于, 所述缓存调整装置装置包括:

比较装置, 用于比较所述存储数据长度和所述可用缓存空间;

缓存扩展装置, 用于根据所述比较装置的比较结果扩展缓存空间的长度;

缓存缩小装置, 用于根据所述比较装置的比较结果缩小缓存空间的长度。

## 一种缓存分配方法及装置

### 技术领域

本发明涉及计算机存储技术领域，具体涉及一种缓存分配方法及装置。

### 背景技术

在通信、计算机编程等领域中，缓存有着广泛的应用。在很多情况下，需要对一些临时数据进行缓存，比如在两个模块之间的数据接口中经常对数据进行缓存，即在存储单元中分配一定的存储空间，用来暂时存储数据。通常采用堆栈或存储队列的方式来实现缓存中数据的存取操作。

目前，采用存储队列实现的缓存系统中，分配缓存的方法通常有以下两种：

一是采用固定分配缓存的方法，即为缓存分配固定的内存空间，在以后的使用中，缓存的大小不发生变化。为了保证每次向缓存中存入数据时不会溢出，就要为缓存分配足够大的空间，因此，就会造成很多情况下存储空间的浪费，若事先分配的缓存空间不是足够大，就会使某些数据丢失。

二是采用动态分配缓存的方法。在计算机中FIFO（先进先出）是一种先进先出的缓存系统，是由一个内存指针来指向的，因此，在每次使用时，根据要存储的数据长度设定指针所指向的内存即可改变缓存的大小。这样虽然可减少存储空间的浪费，但是，在这种方法中，需要频繁地改变一个指针所指向的内存，以满足当前长度数据的存储要求，这样就有可能带来系统的不稳定，同时也降低了缓存系统的效率。

### 发明内容

本发明的目的是提供一种缓存分配方法及装置，以克服现有技术中采用固

定存储空间时的空间浪费及数据溢出的问题。

本发明的目的是通过以下技术方案实现的：

一种缓存分配方法，其特征在于，包括：

A、设定默认缓存空间；

B、获取存储数据长度和可用缓存空间；

C、当所述存储数据长度大于所述可用缓存空间时，按预定扩展规则扩展所述默认缓存空间；

D、当所述存储数据长度小于所述可用缓存空间第一预定值时，按预定缩小规则缩小所述默认缓存空间。

所述默认缓存空间的末端包括至少一个空位。

所述步骤 B 包括：

B1、设定缓存标识参数；

B2、根据所述缓存标识参数的实际值获取所述可用缓存空间。

所述缓存标识参数包括：已用缓存的起始指针、已用缓存的终止指针、缓存满标识、缓存空标识。

所述步骤 B1 包括：

当所述默认缓存空间全部可用时，设定所述缓存满标识为假、缓存空标识为真；

当所述默认缓存空间部分可用时，设定所述缓存满标识为假、缓存空标识为假；

当所述默认缓存空间都不可用时，设定所述缓存满标识为真、缓存空标识为假。

所述步骤 C 中的预定扩展规则包括：

C1、如果所述存储数据长度大于所述可用缓存空间第二预定值，则将所述可用缓存空间扩展为所述存储数据长度；

C2、如果所述存储数据长度小于所述可用缓存空间第二预定值，则将所述可用缓存空间扩展为第二预定值。

所述步骤 D 包括：当所述存储数据长度小于所述可用缓存空间的一半时，将所述可用缓存空间缩小一半。

所述步骤 C 和步骤 D 中存入所述存储数据时，

当所述缓存满标识及所述缓存空标识均为假时，判断所述已用缓存的终止指针是否到达所述默认缓存空间末端的空位前；

如果已到达，则从所述默认缓存空间的起始位置开始依次存入所述存储数据；

如果未到达，则从所述已用缓存的终止指针的下一个位置开始依次存入所述存储数据。

一种缓存分配装置，其特征在于，包括：

参数维护装置，用于维护缓存标识参数；

缓存空间获取装置，用于根据所述参数维护装置中的缓存标识参数获取可用缓存空间；

存储数据获取装置，用于获取存储数据及其长度；

缓存调整装置，用于根据所述缓存空间获取装置和所述存储数据获取装置的输出调整缓存空间的长度；

缓存分配单元，用于依次将所述存储数据存入经过所述缓存调整装置调整后的可用缓存空间。

所述参数维护装置包括：

参数设定装置，用于设定所述缓存标识参数的初始值；

参数更新装置，用于根据所述缓存调整装置的调整结果及所述缓存分配单元的分配结果更新所述缓存标识参数。

所述缓存调整装置包括：

比较装置，用于比较所述存储数据长度和所述可用缓存空间；  
缓存扩展装置，用于根据所述比较装置的比较结果扩展缓存空间的长度；  
缓存缩小装置，用于根据所述比较装置的比较结果缩小缓存空间的长度。

由以上本发明提供的技术方案可以看出，本发明根据存储数据长度相对于可用缓存空间变化的大小来调整缓存空间，对于较大的数据长度，分配较大的存储空间，从而防止了数据的溢出；当存储数据的长度较可用缓存空间小于一预定值时，才改变缓存容量，这样就可以根据系统的实际配置设定调整参数，既避免了缓存容量的浪费，同时又兼顾了改变缓存大小的频率，不需要频繁地改变缓冲区指针的位置，提高了系统的稳定性及数据存取效率。

## 附图说明

- 图 1 是本发明缓存分配方法的详细流程图；
- 图 2 是本发明方法中的默认缓存空间结构的初始状态示意图；
- 图 3 是图 2 所示的缓存空间结构使用部分空间后的一种状态；
- 图 4 是图 2 所示的缓存空间结构使用部分空间后的另一种状态；
- 图 5 是将图 3 所示的缓存空间的可用空间扩展到存储数据长度后的状态；
- 图 6 是将图 3 所示的缓存空间的可用空间扩展一倍后的状态；
- 图 7 是本发明方法中按预定缩小规则缩小缓存空间前后对照图；
- 图 8 是本发明缓存分配装置的结构示意图；
- 图 9 是本发明在 SDH 类逻辑仿真激励数据产生过程中的一个应用实例。

## 具体实施方式

本发明的核心在于在特定的条件下对缓存空间进行调整，即根据存储数据长度与可用缓存空间的相对差值对缓存空间按预定规则进行扩展或缩小：当存储数据长度大于可用缓存空间时，按预定扩展规则扩展缓存空间；当存储数

据长度小于可用缓存空间第一预定值时，按预定缩小规则缩小缓存空间。预定扩展规则包括：如果存储数据长度大于可用缓存空间第二预定值，则将可用缓存空间扩展为存储数据长度；否则将可用缓存空间扩展为第二预定值。这样，使缓存空间始终保持合适容量的同时减少调整的次数。

为了使本技术领域的人员更好地理解本发明方案，下面结合附图和实施方式对本发明作进一步的详细说明。

参照图 1，图 1 描述了本发明缓存分配方法的详细流程，包括以下步骤：

步骤 101：设定默认缓存空间，也就是分配一定长度的计算机内存，用于存储数据。

默认缓存空间结构的初始状态如图 2 所示：

该缓存空间的默认长度为 DefaultLength；

在默认缓存空间的末端设定一个或多个空位，在为存储数据分配缓存空间时，该空位始终不被使用，以防止系统出现异常情况时数据溢出。在该例中包括一个空位。

步骤 102：设定缓存标识参数。所述缓存标识参数包括：已用缓存的起始指针、已用缓存的终止指针、当前指针、缓存满标识、缓存空标识。其中，所述当前指针用来指示当前数据的位置；所述缓存满标识（FullFlag）表示默认缓存空间中是否已没有可用空间，所述缓存空标识（EmptyFlag）表示默认缓存空间中是否还有可用空间，因此，缓存满标识和缓存空标识有以下三种组合状态：

当默认缓存空间全部可用时，缓存满标识为假、缓存空标识为真；

当默认缓存空间部分可用时，缓存满标识为假、缓存空标识为假；

当默认缓存空间都不可用时，缓存满标识为真、缓存空标识为假。

在图 2 所示的缓存空间结构的初始状态中，已用缓存的起始指针（StartP）和已用缓存的终止指针（EndP）均指向设定的默认缓存空间的始端，即

StartP=0, EndP=0, 当前数据指针 Pointer=0, 此时,

默认缓存空间的总容量 AllCapacity=DefaultLength;

已用空间 UsedCapacity=0;

可用空间 EmptyCapacity=DefaultLength-1;

缓存满标识 FullFlag=false; 缓存空标识 EmptyFlag=true。

图 3 和图 4 是图 2 所示的缓存空间结构使用部分空间后的两种状态:

默认缓存空间使用部分空间后, 已用缓存的起始指针在默认缓存空间的起始位置, 即 StartP=0 时, 缓存空间结构的状态如图 3 所示。此时,

默认缓存空间的总容量 AllCapacity=DefaultLength;

已用空间 UsedCapacity=EndP;

可用空间 EmptyCapacity=DefaultLength-1-EndP;

缓存满标识 FullFlag=false; 缓存空标识 EmptyFlag=false。

默认缓存空间使用部分空间后, 已用缓存的起始指针在默认缓存空间的中间位置, 即 StartP>0 时, 缓存空间结构的状态如图 4 所示。此时,

默认缓存空间的总容量 AllCapacity=DefaultLength;

已用空间 UsedCapacity=EndP-StartP;

可用空间 EmptyCapacity=DefaultLength-1-(EndP-StartP);

缓存满标识 FullFlag=false; 缓存空标识 EmptyFlag=true。

步骤 103: 根据缓存标识参数的实际值获取可用缓存空间。

对于已用缓存的起始指针在默认缓存空间的起始位置的不同, 有以下两种情况:

当 StartP=0 时, 可用缓存空间 EmptyCapacity=DefaultLength-EndP;

当 StartP > 0 时, 可用缓存空间

EmptyCapacity=DefaultLength-(EndP-StartP)。

步骤 104: 获取存储数据长度。

步骤 105: 判断存储数据长度是否大于可用缓存空间。

如果存储数据长度大于可用缓存空间, 则进到步骤 106: 进一步判断存储数据长度是否大于可用缓存空间第二预定值。比如, 可以设定第二预定值为 2 倍可用缓存空间。

如果是, 则进到步骤 107: 将可用缓存空间扩展为存储数据长度。然后, 进到步骤 111: 存入存储数据。

如果不是, 则进到步骤 108: 将可用缓存空间扩展为第二预定值。然后, 进到步骤 111: 存入存储数据。

如果存储数据长度小于可用缓存空间, 则进到步骤 109: 进一步判断存储数据长度是否小于可用缓存空间第一预定值。比如, 可以设定第一预定值为二分之一可用缓存空间。

如果是, 则进到步骤 110: 按预定缩小规则缩小默认缓存空间。然后, 进到步骤 111: 存入存储数据。

如果不是, 则不调整可用缓存空间的大小, 直接进到步骤 111: 存入存储数据。

步骤 112: 更新缓存标识参数。

图 3 所示的缓存空间按照本发明方法中的缓存扩展规则扩展后的缓存空间状态分别如图 5 和图 6 所示。

参照图 5, 图 5 是将图 3 所示的缓存空间的可用空间扩展到存储数据长度后的状态:

假设扩展前的可用空间为  $N$ , 扩展的长度为  $M$  ( $M > N$ ), 则此时:

默认缓存空间的总容量  $AllCapacity = DefaultLength + M$ ;

已用空间  $UsedCapacity = EndP$ ;

可用空间  $EmptyCapacity = 0$ 。

缓存满标识 FullFlag=true; 缓存空标识 EmptyFlag=false。

再参照图 6, 图 6 是将图 3 所示的缓存空间的可用空间扩展一倍后的状态:

假设扩展前的可用空间为 N, 则先将可用空间扩大一倍, 然后将数据存入可用空间内, 此时, 已用缓存的终止指针 EndP 更新为原 EndP+存储数据长度。将可用空间的大小减去存入数据长度, 再检查其大小, 如果为 0, 则 FullFlag 信号置为 True, 其它参数保持不变。

默认缓存空间的总容量 AllCapacity=DefaultLength+N;

已用空间 UsedCapacity=EndP;

可用空间 EmptyCapacity=DefaultLength+N-1-EndP。

缓存满标识 FullFlag=false; 缓存空标识 EmptyFlag=false。

图 7 示出了按照本发明方法中缓存缩小规则 (设定第一预定值为二分之一可用缓存空间时) 缩小缓存空间前后对照图:

先将缩小前的可用空间 EmptyCapacity 减小一半, 并释放掉相应的内存空间, 存入数据后 EmptyCapacity 减去存入数据长度, 然后检查 EmptyCapacity 是否为 0, 如是则 FullFlag 为 true, 其他信号保持不变。

假设缩小前的可用空间为 N, 缩小可用空间一半并存入数据后, 则此时:

默认缓存空间的总容量 AllCapacity=DefaultLength+M-N/2;

已用空间 UsedCapacity=EndP;

可用空间 EmptyCapacity= DefaultLength+M-N/2- EndP。

缓存满标识 FullFlag= false; 缓存空标识 EmptyFlag=false。

为了提高缓存空间的利用率, 在本发明方法中, 对于缓存空间采用循环利用的方式, 即已用缓存的终止指针到达默认缓存空间末端的空位前时, 如果缓存空间未滿, 则接着使用始端的空间。具体为:

当缓存满标识及缓存空标识均为假时，判断已用缓存的终止指针是否到达默认缓存空间末端的空位前；

如果已到达，则从默认缓存空间的起始位置开始依次存入存储数据；

如果未到达，则从已用缓存的终止指针的下一个位置开始依次存入存储数据。

本发明还提供了对默认缓存空间的多种操作，比如，通过当前数据指针 Pointer 从指定位置取出一定量的数据、删除缓存空间中指定位置的一定量的数据、清空缓存空间等。同样，在对缓存空间中部分或全部进行删除后，需要更新对应的缓存标识参数。

下面参照图 8 所示的本发明缓存分配装置的结构示意图，详细说明本发明装置的工作过程：

首先由参数维护装置 81 中的参数设定装置 811 设定缓存标识参数的初始值，所述缓存标识参数用来标识缓存空间的状态，比如可以包括：存储单元 85 的长度、已用缓存的起始指针、已用缓存的终止指针、当前指针、缓存满标识、缓存空标识等；然后，由缓存空间获取装置 82 根据参数维护装置 81 中的缓存标识参数获取可用缓存空间；同时由存储数据获取装置 83 获取存储数据及其长度；然后，分别将获取的可用缓存空间及存储数据长度传送给缓存调整装置 84 中的比较装置 841，由比较装置 841 按照预定规则比较存储数据长度和可用缓存空间的大小，然后，根据比较结果通知缓存扩展装置 842 或缓存缩小装置 843 将存储单元 85 扩展或缩小到一预定值。

比如按照如下规则：

如果存储数据长度大于可用缓存空间一预定值，则通知缓存扩展装置 842 将存储单元 85 中的可用缓存空间扩展为存储数据长度；否则通知缓存扩展装置 842 将其扩展为一预定长度；

如果存储数据长度小于可用缓存空间一预定值，则通知缓存缩小装置 843 将存储单元 85 中的可用缓存空间缩小为一预定长度，否则不作调整。

同时，缓存扩展装置 842 及缓存缩小装置 843 还要将对存储单元 85 的调整结果通知参数维护装置 81 中的参数更新装置 812，参数更新装置 812 根据该通知更新参数设定装置 811 中的缓存标识参数。

通过上述对存储单元 85 的调整，使得存储单元 85 的可用空间总是适应存储数据的长度，既不会造成数据的溢出，也不会对缓存空间造成过多的浪费。而且对存储单元的调整是在一定条件下进行的，不需要在每次存储新的数据时都进行设定，保持了存储单元的相对稳定性，从而提高了整个系统工作的稳定性。

对存储单元 85 中的可用空间调整到适应存储数据的长度后，由缓存分配单元 86 将存储数据获取装置 83 中获取的存储数据依次存入存储单元 85 中。同时，通知参数更新装置 812 根据存储数据长度更新参数设定装置 811 中的缓存标识参数。

当需要清除存储单元 85 中的部分或全部数据时，通过缓存清除单元 87 根据清除命令清除指定单元的数据。同时，通知参数更新装置 812 根据清除命令中数据长度及数据起始位置更新参数设定装置 811 中的缓存标识参数。

为了使本技术领域的人员更好地理解本发明方案，下面描述本发明的一个具体应用实例，如图 9 所示：

该例为在 SDH 类逻辑仿真激励数据产生装置中从报文生成模块到 LAPS（链路访问过程-SDH）封装模块之间使用缓存的例子。在该例中，本发明的数据具体为报文，为了叙述方便，有时也称作报文数据。

在该例中，报文生成模块用于产生报文数据；LAPS 封装模块用于对报文生成模块产生的报文数据进行封装；缓存管理模块，用于对缓存进

行管理，同时与报文生成模块和 LAPS 封装模块进行通讯，并保证报文生成模块和 LAPS 封装模块协同工作；FIFO 是用于存放报文数据的缓存。缓存的使用情况由缓存的状态来标识，缓存的状态包括：缓存空间总容量、可用空间、已用空间、缓存满标识、缓存空标识、起始指针及终止指针。

根据本发明的缓存分配过程如下：

LAPS 封装模块从缓存 FIFO 中取报文数据进行封装，取数据时先检查缓存 FIFO 的状态，如果发现缓存 FIFO 为空，则通知缓存管理模块，然后缓存管理模块通知报文生成模块产生报文数据，报文生成模块根据配置参数产生报文后放入缓存 FIFO 中，然后 LAPS 封装模块从缓存中取得报文数据进行封装。

由于报文生成模块产生的报文长度变化情况决定着缓存 FIFO 的使用，下面以报文长度依次为 30 字节、40 字节（C 类报文）、50 字节为例来说明缓存的状态。

工作过程如下：

（1）假设在初始状态的缓存状态如下表 1 所示（假设缓存的默认长度为 100，即 DefaultLength=100），LAPS 封装模块从缓存中取报文进行封装，经检查发现缓存是空的，则通知缓存管理模块；

表 1：

状态名	值	状态名	值
缓存空间总容量 AllCapacity	100		
可用空间 EmptyCapacity	100	已用空间 UsedCapacity	0
缓存满标识 FullFlag	False	缓存空标识 EmptyFlag	True
起始指针 StartP	0	终止指针 EndP	0

(2) 缓存管理模块通知报文生成模块，要求其开始工作产生报文；

(3) 报文生成模块根据报文配置参数生成第一个报文，报文长度为30字节，按照报文的结构产生一个30字节的报文放入缓存中，在往缓存中存放时先检查缓存的状态，发现缓存为空且容量为100，经判断，发现将缓存缩小一预定值（在本例中假设为1/2）后还能够存放该报文数据，所以，需要缩小缓存，即将缓存的可用空间缩小一半再将报文数据放入缓存中，此时缓存的状态如下表2所示：

表2:

状态名	值	状态名	值
缓存空间总容量 AllCapacity	50		
可用空间 EmptyCapacity	20	已用空间 UsedCapacity	30
缓存满标识 FullFlag	False	缓存空标识 EmptyFlag	False
起始指针 StartP	0	终止指针 EndP	30

(4) LAPS封装模块从缓存中取得报文数据并更新缓存的状态，以备存放下一报文，此时缓存的状态如下表3所示，然后按LAPS协议进行封装报文；

表3:

状态名	值	状态名	值
缓存空间总容量 AllCapacity	50		
可用空间 EmptyCapacity	50	已用空间 UsedCapacity	0
缓存满标识 FullFlag	False	缓存空标识 EmptyFlag	True
起始指针 StartP	0	终止指针 EndP	0

(5) LAPS封装模块进行第二个报文的封装，取数据时先检查缓存的状态，

如果发现缓存空标识为True，则通知缓存管理模块，然后缓存管理模块通知报文生成模块产生报文数据，报文生成模块根据配置参数产生报文后放入缓存中，然后LAPS封装模块从缓存中取得报文数据进行封装。报文生成模块生成第二个报文（假设报文长度为40字节）放入缓存时，先检查缓存的状态，发现缓存的可用空间能够满足要求，且不需要伸缩缓存，可以将报文数据直接存入缓存，此时的状态如下表4所示：

表4:

状态名	值	状态名	值
缓存空间总容量 AllCapacity	50		
可用空间 EmptyCapacity	10	已用空间 UsedCapacity	40
缓存满标识 FullFlag	False	缓存空标识 EmptyFlag	False
起始指针 StartP	0	终止指针 EndP	40

(6) LAPS封装模块取走缓存中的报文数据并更新缓存的状态，此时的缓存状态如下表5所示：

表5:

状态名	值	状态名	值
缓存空间总容量 AllCapacity	50		
可用空间 EmptyCapacity	50	已用空间 UsedCapacity	0
缓存满标识 FullFlag	False	缓存空标识 EmptyFlag	True
起始指针 StartP	0	终止指针 EndP	0

(7) LAPS封装模块进行第三个报文（50字节的报文），报文生成模块产生报文放入缓存时检查其状态，仍然发现缓存满足要求而不收缩缓存，故可直接存入报文数据，此时缓存的状态如下表6所示：

表6:

状态名	值	状态名	值
缓存空间总容量 AllCapacity	50		
可用空间 EmptyCapacity	0	已用空间 UsedCapacity	50
缓存满标识 FullFlag	True	缓存空标识 EmptyFlag	False
起始指针 StartP	0	终止指针 EndP	50

(8) LAPS封装模块从缓存中取走50字节的报文数据后清空缓存, 然后进行LAPS封装, 此时缓存的状态如下表7所示:

表7:

状态名	值	状态名	值
缓存空间总容量 AllCapacity	50		
可用空间 EmptyCapacity	50	已用空间 UsedCapacity	0
缓存满标识 FullFlag	False	缓存空标识 EmptyFlag	True
起始指针 StartP	0	终止指针 EndP	0

(9) LAPS封装模块进行第四个报文(假设报文长度为70字节), 报文生成模块产生报文放入缓存时检查其状态, 经检查, 发现可用空间为50, 则进一步判断报文长度是否大于可用空间一预定值(在本例中假设预定值为2倍可用空间), 则将缓存扩展一预定值(在本例中将缓存扩展可用空间的大小, 即将可用空间扩大一倍), 扩展内存的过程如下: 先建立一个临时缓存, 接着将当前缓存中的数据备份到临时缓存中, 然后释放当前缓存的内存空间, 接着按本发明所确定的缓存空间大小, 为当前缓存分配内存空间, 再将临时缓存中的数据复制到新建立的缓存中, 最后释放临时缓存的内存空间并更新缓存的状态。然后将报文存入缓存中, 并更新缓存的状态, 此时缓存的状态如下表8所示:

表8:

状态名	值	状态名	值
缓存空间总容量 AllCapacity	100		
可用空间 EmptyCapacity	30	已用空间 UsedCapacity	70
缓存满标识 FullFlag	False	缓存空标识 EmptyFlag	False
起始头指针 StartP	0	终止指针 EndP	70

(10) LAPS封装模块从缓存中取走70字节的报文数据后清空缓存, 然后进行LAPS封装, 此时缓存的状态如下表9所示:

表9:

状态名	值	状态名	值
缓存空间总容量 AllCapacity	100		
可用空间 EmptyCapacity	100	已用空间 UsedCapacity	0
缓存满标识 FullFlag	False	缓存空标识 EmptyFlag	True
起始指针 StartP	0	终止指针 EndP	0

上述例子在使用自动伸缩的缓存时每次用完后都清空缓存中的数据。在实际应用中, 缓存中的数据可能不是即时清空, 而是在一定条件下清空。在这种情况下对缓存空间的维护与上面类似, 在此不再详细描述。

从上述例子可以看出, 在缓存的使用过程中, 其大小总是在尽量适应被存入数据的长度, 而又不会频繁的去改变内存的大小, 一般情况下只是在第一次使用缓存的大小 (让缓存的大小尽量适应存放数据的长度), 只要以后存入的数据长度之间相差不到一倍的大小, 缓存就不需要自动伸缩, 这样既有效地节约了内存空间, 又不需要频繁地去操作内存空间 (申请和释放), 提高了程序的可靠性; 如果在此过程中使用一般的动态缓存, 每次都需要改变缓存的大小

以适应存入数据的长度，这样虽然在一定程序上节省了内存空间，但频繁地操作内存可能会带来程序的不稳定，如果使用静态缓存（即缓存大小固定），则在存入数据长度很小时会有较大的内存空间浪费。

在本发明中，由于采用动态分配存储空间，对于较大的数据长度，缓存分配单元会分配较大的存储空间，从而防止了数据的溢出。

同时，由于缓存分配单元根据数据长度与可用缓存容量是否小于一预定值，来决定是否将缓存缩小一预定值，这样就会使可用缓存容量不会超过数据长度太多，从而不会太多地浪费存储空间。

再者，当数据长度较可用缓存容量小于某一特定值范围时，就不改变缓存容量，与现有的动态分配缓存技术相比，减小了改变缓存大小的频率，因而提高了系统的效率。

虽然通过实施例描绘了本发明，本领域普通技术人员知道，本发明有许多变形和变化而不脱离本发明的精神，希望所附的权利要求包括这些变形和变化而不脱离本发明的精神。

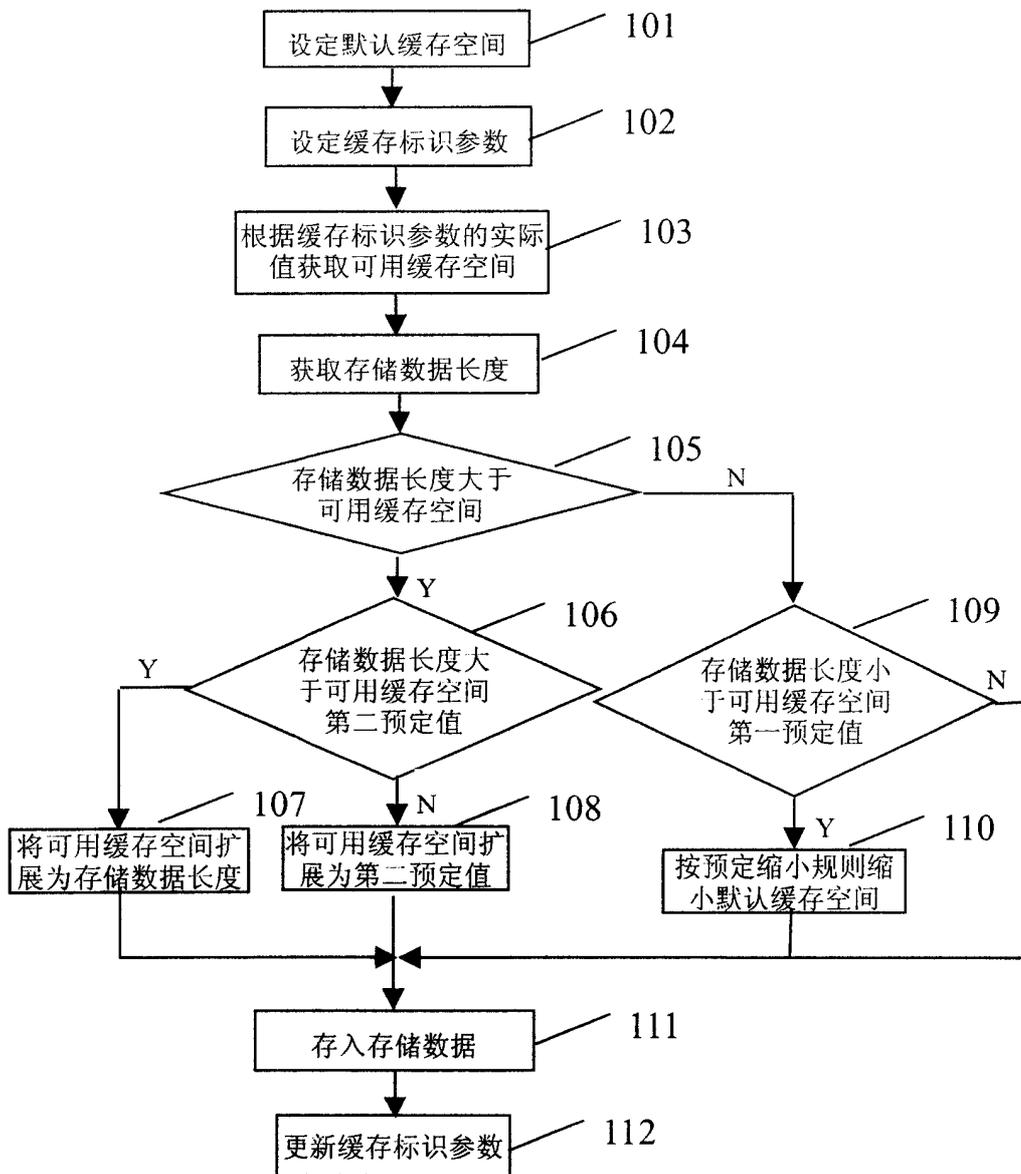


图 1

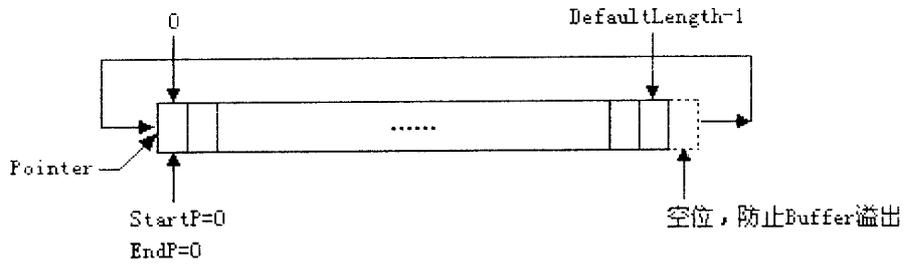


图 2

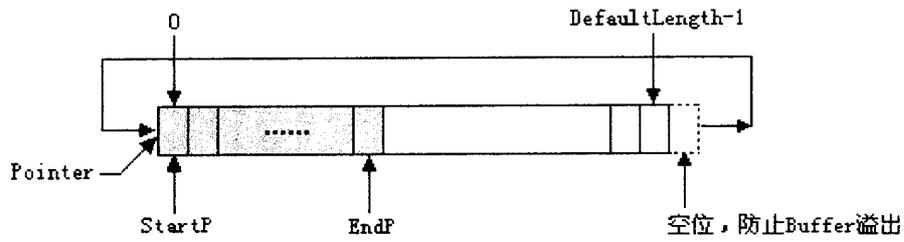


图 3

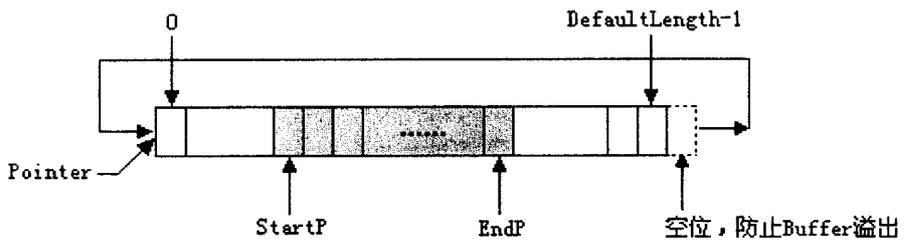


图 4

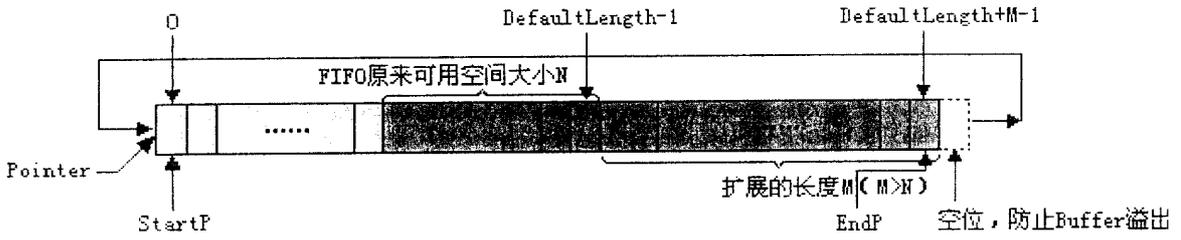


图 5

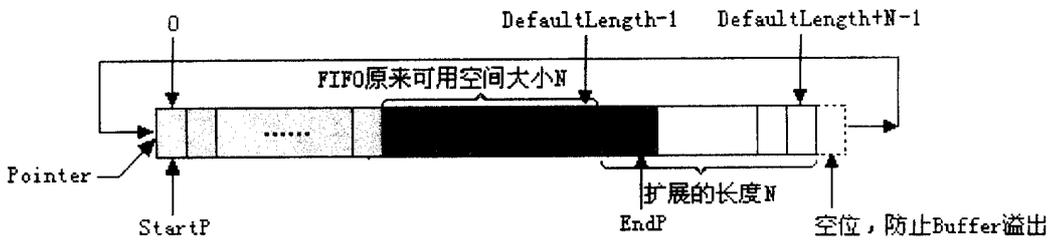


图 6

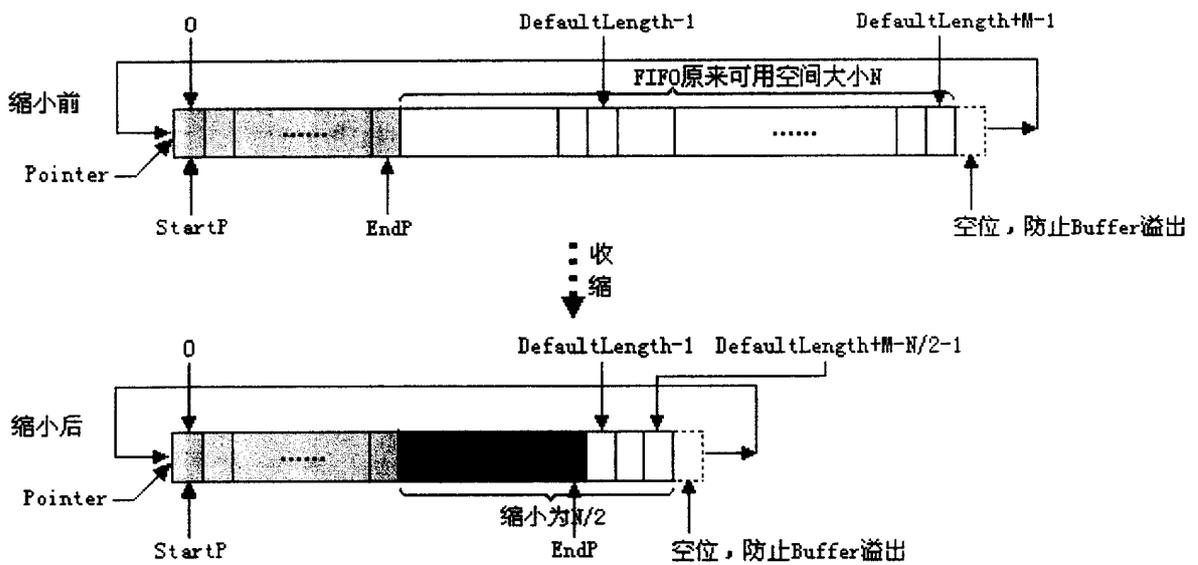


图 7

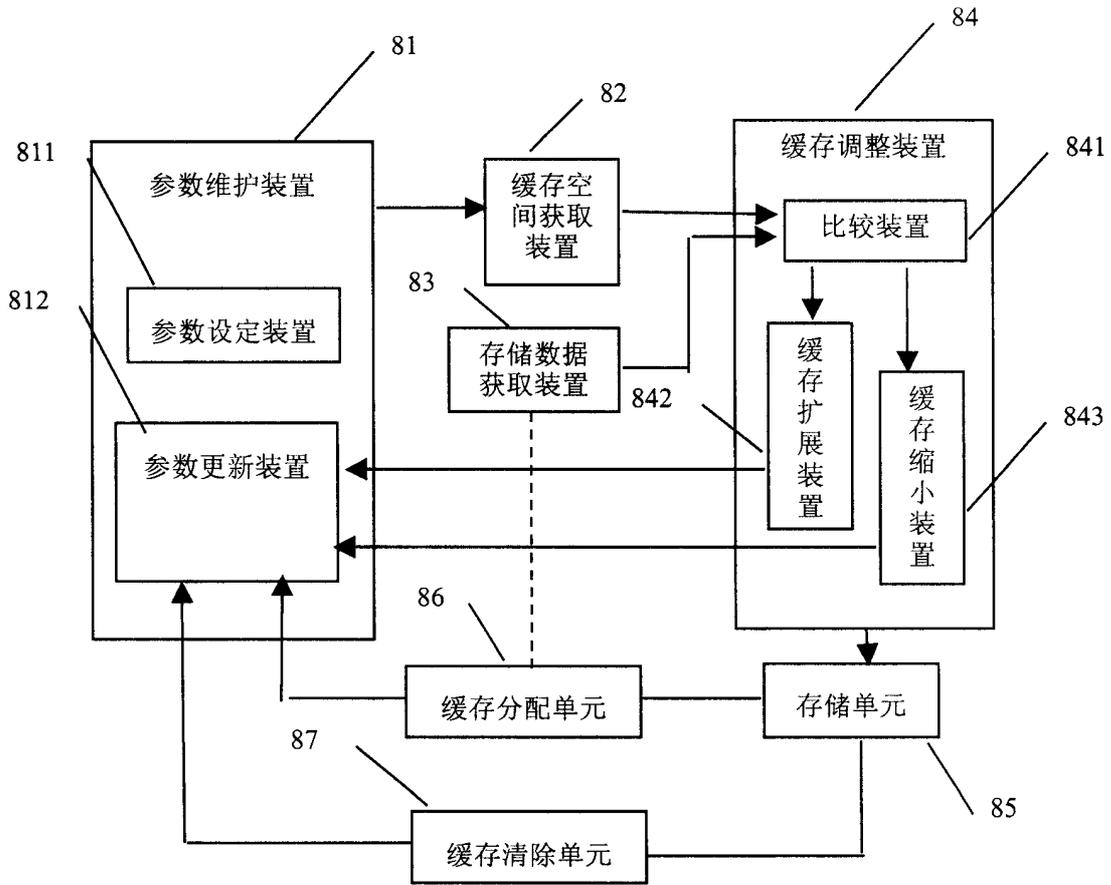


图 8

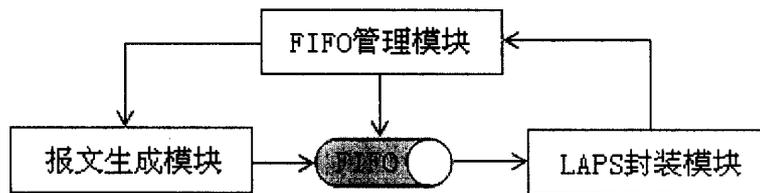


图 9