



166166

A DATA UPDATING METHOD BASED ON DATA BLOCK COMPARISON

FIELD OF THE INVENTION

- 5 This invention relates to the computer technical domain, especially involving a data updating method.

BACKGROUND OF THE INVENTION

- 10 The number of popular cyber games in China is over 100, and almost every day, there are more than 30% of cyber games that have their contents updated. The games in the computers of the Internet bar also require timely updating to satisfy the needs of the netizens in the cybercafé. There are two technical difficulties in solving this demand. One is that the number of the terminals of the machines in all cybercafés is
- 15 huge, with the approximate number of 10 million sets. If the games are all updated directly from the Internet servers, the performance load of the server cannot handle at all. Meanwhile, the server also needs to distribute huge network bandwidth on the Internet to support the updating; the other difficulty is that the updating amount of the popular games is tremendous. If viewed from the updating level of files, it is
- 20 about 40G on average every day at present. The huge updating amount almost makes the game updating incapable of normal operation in the Internet bar basically. The bandwidth of the Internet bar is limited, and thus the time required for the game updating is unbearably long.
- 25 In order to solve the above problems, the approach of data increment update is also adopted in the existing technology, i.e. inspect the difference between data before and after updating. When updating, only this different part is updated. Generally, the logic file is taken as the object for atomic processing. The comparison of the updated difference takes the attribute of the file itself like the time for file creation or the time
- 30 for file modification as the basis to determine whether this file has been updated. If



G00002

the file has been updated, the whole file is taken as the difference demand, and the network transmission is required to achieve the goal of data updating.

However, the existing method takes the logic file as the processing object. In the actual application, the updating of many data only changes a segment of the data in the file, which causes the oversize of the granularity for the updated difference comparison. As a result, some original data will be repeatedly updated.

If the file compared from the updated difference requires updating, the length of the file is the size requiring updating. Since the size of the file depends, the atomic updating length is thus dynamically changing, which will cause the reduction of the network transmission performance.

SUMMARY OF THE INVENTION

This invention has provided a quick data updating method. Through the division of data and specific updating algorithm, the speed for data updating has been greatly improved. In addition, the load of the server is further relieved with the improved network downloading architecture, and the network transmission performance is enhanced.

The data updating method of this invention is realized through the fragment index and difference comparison of the data content. View a complete set of data (e.g. all data in a game) as a continuous and complete binary data flow. A file is a logic unit in a segment of binary data flow. A data block is also a logic unit of the binary data flow.

The relationship of the three: a data block is the subset of a file; a file is the subset of a game data. When the overall size of the file is smaller than the defined length of a data block, the content of a file is a data block. Certainly, that a file contains several data blocks is often the case. The length of a data block can be manually defined, and thus we are able to find out the data block length with best overall performance in the global system according to the characteristics of the network transmission and disk

(currently, the disk array is abundantly used) buffer instead of passively adapting to the file length.

A data updating method based on data block comparison includes the following steps:

5

(1) Divide all the version data into several data blocks. Extract the following items of this version data specific to each version data:

a) Attribute and identity mapping information of the overall version data;

b) Attribute and identity mapping information of all files;

10 c) Identity mapping information of each data block;

(2) Create corresponding index files specific to each version data. The said index file includes the information extracted from Step I of this version data;

15 (3) When the data is being updated, compare the index file in the previous version data and the later version data. When comparing, the steps are as follows:

a) Find out the excluded files in previous version data and all changed files according the attribute and identity mapping information of all files in the later version index file;

20 As for the excluded files in the previous version data, update directly. In other words, copy this file in the later version data to the previous version data;

As for the changed files, find out changed data block in the previous and later version data according to the identity mapping information of each data block in the file, and update the changed data blocks. In other words, copy this data block in the

25 later version data to the previous version data;

b) Find out excluded files in the later version data and changed files according to the attribute and identity mapping information of all files in the previous version index files;

As for the excluded files in the later version data, delete them directly;

30 As for the changed file, find out excluded data blocks in the later version data according to identity mapping information of each data block in the file, and delete this data block directly.

In the Step I of this invention, when dividing a certain version data into several data blocks, a single file is taken as the object, and the files are divided one by one. The data in different files will not be divided into a same data block since the length of the data block is pre-specified. When a file is divided into the end, the length of the
5 end block is not generally exactly equal to the length of the pre-specified data block. So, the processing method for the end block of a file is as follows:

- One thing to point out is that the length of the end block must be less than or equal to the length of the pre-specified data block; otherwise, it will be continuously divided.
- 10 When the length of the end block is equal to that of the pre-specified data block, divide it according to the preset principle. What we are considering with emphasis is the case that the length of the end block is less than that of the pre-specified data block.
- 15 First, set up a threshold value and calculate the length of the end block to work out the ratio between the length of the end block and the length of the data block pre-specified. When the said ratio is smaller than threshold value, combine the end block to the previous data block; if the ratio is greater than or equal to the threshold, divide this end block to a data block separately. As for preference, take the threshold value
20 as $1/3$, which can reduce the sporadic short-length data block as much as possible to benefit the improvement of the network transmission efficiency and also avoid frequent data re-write to the greatest extent.

The length of the data block has a direct effect on the updating of the data:
25

If the length of the data block is oversized, the distortion will occur during the network transmission to cause the resending of the whole data block, during which abundant network resources and machine performance resources will be occupied, thus causing a lot of trouble. It is also a shortcoming for the simple processing
30 method.

If the data block is moderately defined, massive data can be obtained and assembled from several servers since one data block can be obtained from several servers, and thus the whole network resources will be equally assigned to the bandwidth of each

server. As the length of the preferred data block, the adoption of 16~64K is relatively proper. 32K is adopted or the length of the further preferred data block, at the moment of which the consideration given to both network transmission quality and program performance is achieved.

- 5 If the data block is defined too small, the number of times for interaction between the program and the operating system has been increased, thus influencing the program performance.

There are many methods in the existing technology for obtaining the identity mapping information of data block (a segment of binary flow), such as MD5 code and CRC32.

Comparing MD5 code and CRC32, we can see the collision rate of MD5 code is much lower, which is to say MD5 code is able to different between two different segments of binary data flow while CRC32 may consider these two segments as the same. There are only four bytes in the identity information produced by CRC32. However, 128 bytes may be operated in the identity information produced through MD5. The time complexity between the two is not large.

20 No matter which method is used to build the identity mapping information of the data block, the purpose is to compare the identity mapping information with relatively small data amount to replace the comparison content of the data block. Assume the length of the data block is 32K bytes, and we will use 4 bytes or 128 bytes to compare to replace the comparison of 32K bytes.

25 This invention finally selects CRC32 as the identity mapping information. The main consideration is that the hit rate of CRC32 is acceptable, and also the identity mapping information occupies small space, which is good for the reduction of the size of the index.

30 The index file said in Step II is a set which records the identity mapping information of all the data blocks of a version data. The data of different version must correspond to the index file of a version.

The function of the index file is as follows: the program requiring updating generates difference demand amount through the comparison of different versions of index files. Thus, the program requiring updating only needs to download the data of the different part to finish the upgrading of the version.

5 The content in the index file of this invention includes the following items:

a) The attribute and identity mapping information of the overall version data, including:

PkgId	Type identity of the version data. Different types of version data adopt different PkgId. For different games, the PkgId of the version data is different;
BlockSize	Unit size of the file block (unit byte), i.e. the length of the data block
PkgIdxVer	Index version, identity of version data of the same type but different version data. For example, a same game has different version, and then PkgId of different version is the same, but PkgIdxVer is different.
SelfCRC	The self-check of CRC in the index file is used to examine whether the index file itself is correctly transmitted during the network transmission process;
FileDescList	List for the file to describe the information (several files describe the array of the information). It shows the attribute information and identity mapping information of all the files in this version data.

10

b) FileDescList, the list for the file to describe the information, includes the attribute and identity mapping information of all files in this version data. As far as each file is concerned, the description information (FileDescInfo) is as follows:

FileAttr	File type, including directory and file;
FilePathName	Relative path included in the file name;
FileModifyTime	Time for file modification;
FileCRC	CRC check code of the whole content of the file;

FileLength	File length (unit byte);
BeginBlockId	Number of the first data in the file; as far as the entire version data is concerned, the numbers of all the data blocks are unique. Through the number of data block, the position of the data block in the entire version data and the whole file can be located quickly;
BlockDescList	List of file data block information (array of several data block information). It shows the identity mapping information of all data blocks in this file;

c) BlockDescList, the file data block information list, includes the identity mapping information of all data blocks in this file. As far as the data block is concerned, its description information (BlockDescInfo) is as follows:

5

BlockCRC	CRC check code of the content of the data block.
----------	--

In Step III, when comparing the previous version data and the later version data, a difference comparison algorithm based on fragment index has been adopted:

10

There are two major steps:

I. After obtaining the index files in two versions, first take the index file of the new version (later version data) as the reference to traverse the index file of the new version and take out the file description information (FileDescInfo) in the file type to search the information of this file in the index file of the old version (previous version data). If it cannot be found, it shows that this file is newly added, and the difference comparison result is generated (i.e. all this file needs updating). If it is found, determine if the FileCRC of FileDescInfo of the two indexes are the same. If yes, we think this file has not been changed; if not, we have to search for the changed data blocks, and begin to traverse the block description information (BlockDescInfo) of this file according to the order. If the BlockCRC corresponding to the new and old version is different, we conclude that this data block requires updating.

25 II. The above operation can process the effects of the new addition and modification

of the data block. If the file of the new version has deleted the data block, the processing method is as follows:

5 Take the index file of the old version as the reference, and traverse the index file of the old version. Take out the file description information in the file type to search in the index file of the new version. If it cannot be found, it shows this file will be deleted.

As for the block deletion, the procedures are the same as the foregoing.

10

The differences generated from the above two steps have two types: one requiring downloading and updating and the other requiring replacement. It reflects which kind of data shall be downloaded, and where the updated data blocks shall be placed when assembling with the data.

15

However, there is the information of BeginBlockId (the serial number of the first block of the file) in the file information description in the index file, and thus when traverse the array of the file information description, the position of a give data block stored in the disk can be preciously determined according to calculation of the length of the data block and the offset.

20

In Step III of this invention, when updating the data block, it is actually carried out between the supplier end of the data block and the request end of the data block. In other words, the later version data is stored in the supplier end of the data block, while the previous version data is stored in the request end of the data block. When updating the data, the following method is adopted in order to improve the throughput and transmission speed of the data volume between the supplier end and request end of the data block:

25

30 (1) Create a shared cache (DownCache) in the request end of the data block. When updating the data, the supplier end of the data block read the specified data block in the shared cache according to the request of the request end of the data block; create shared cache to realize uniform management of all the requests in the client (request end of data block).

(2) Upon the reception of the downloading application of the data block, the supplier end of the data block first read the shared cache. If this data block is in the shared cache, transmit the requested data block to the request end of the data block immediately;

If this data block is not in the shared cache, the reading of file is not carried out immediately. Instead, the downloading application of this data block will be submitted to the request queue.

10

(3) When the supplier ends of the data block are idle, process the downloading application in the request queue according to the sequence of the time for the submission of the downloading application, and transmit the requested data block to the request end of the data block.

15

As for preference, sort the data blocks in the shared cache according to the hit rate (the hit rate is the number of times for reading the same block). When the shared cache is full, the data block at the end will be eliminated from the shared cache.

20 In order to avoid that the downloading application of a certain data block waits too long in the request queue, the threshold of the waiting time is set. When the waiting time of the downloading application of a certain data block in the request queue exceeds the threshold value, the downloading application of this data block will be processed with priority even if the supplier end of the data block is not idle.

25 Generally speaking, the time threshold can be set to 15 to 30 minutes.

In order to further reduce the time for track seeking when reading and writing the file, carry out ascending sort according to the serial number for the data blocks with the same hit rate in the shared cache.

30

The technical terms mentioned in this invention can be interpreted as follows if no special instructions are provided:

1. Data block: it is a set of binary data flow, describing the granularity of the network or logic processing.

2. Index file: the data block describes a complete set of game data set. Produce identity mapping information to each data block. The file which records the identity mapping information of these data blocks and part of game attribute information is called as index file.
3. Incremental update: focus on a class of operation with the version upgrading attribute. By relying on the original data, only a small amount of data is required to update, and the purpose of upgrading can be achieved.
4. IDC (Internet Data Center): Internet Data Center.
5. CRC32 (Cyclic Redundancy Check): 32 bit cyclic redundancy check code.
6. Time complexity: refer to the time resource consumed by the algorithm. If the scale of one problem is n , the time required for solving this problem through a certain algorithm is $T(n)$. It is a certain function $T(n)$ of n , which is called the "time complexity" of this algorithm.
7. Space complexity: refer to the occupied storage space required to finish an algorithm in the computer science domain.
8. MD5 code (message-digest algorithm 5): information-abstract algorithm. It is widely used in the encryption and decryption technologies. It can be called the "digital fingerprint" of file.
9. Compilation: process of centralizing the discretely distributed data to continuous data.
10. Decompilation: reverse operation of compilation.

The data updating method of this invention has greatly improved the data updating speed through the division of data and specific updating algorithm. In addition, the load of the server is further relieved with the improved network downloading architecture, and the network transmission performance is enhanced.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is the structural schematic diagram of the index file of this invention;

Figure 2 is the schematic diagram of the flow of the difference comparison algorithm of this invention;

5 Figure 3 is the frame diagram of overall system utilizing the data updating method of this invention;

Figure 4 is the frame diagram for the updating passage of game data;

Figure 5 is the schematic diagram of newly added game within IDC layer;

Figure 6 shows the data updating flow of IDC layer and Internet bar service layer;

10 Figure 7 shows the data updating and program interaction of Internet bar server layer and Internet bar computer layer;

Figure 8 is the flow chart for the optimization of cache downloading;

Figure 9 is the schematic diagram for TCP2P business program interaction.

15

DETAILED DESCRIPTION OF THE INVENTION

The data updating method of this invention is explained with the combination of the specific hardware and network resource. Take the updating of the game data as the example.

20

This invention updates the game updating to the final updating of the Internet bar from IDC layer to terminal layer through the establishment of three-layer updating method (the computer of the Internet bar is the terminal layer, the server of the Internet bar is the bar layer, and the Internet server is IDC layer). Through the three-layer updating method, the updating volume at the terminal is collected to the bar layer so as to effectively reduce the updating volume and solve the problem of the concurrency choke point of abundant terminals and a huge updating volume. This three-layer architecture is shown in Figure 3.

25

30 Where:

Terminal layer: the program name is BarClient; the operating object is the netizen. It has the function of penetration restoration and game updating. This program only communicates with the bar server.

Bar layer: the program name is BarServer; the operating object is the webmaster or

the owner of the Internet bar. It has the function of BarClient end updating, updating game data from IDC and webmaster.

IDC layer: provide the making of the latest game version and supply BarServer to download the game.

5 Refer to Figure 4. The program and its architecture in IDC layer include:

1. Game making tool (PMUpdate): the function of this program is to carry out compilation on the game data and make relevant indexes. In other words, as for a same game program, it generates different version data, and divides all the version data to several data blocks (the preset length of each data block is around 32k).

10

The following processing is carried out specific to the end block of a file:

```
if (nFileLen % nBlockSize < nBlockSize / MagicBlockGene) { -- nBlockCnt; }
```

```
nBlockCnt=Number of blocks;
```

```
nFileLen=File length;
```

15 nBlockSize=Unit size of file segment;

```
nBlockCnt = nFileLen/ nBlockSize + 1
```

```
if (nFileLen % nBlockSize < nBlockSize / MagicBlockGene) { -- nBlockCnt; }
```

20 MagicBlockGene (with the value of 1/3) is a parameter set for the purpose of saving the storage space. It can effectively prevent the occurrence of the condition that the end block is too small and occupies the size of the complete block. When the end block is too small (i.e. smaller than $nBlockSize / MagicBlockGene$), and is combined to the previous block instead of forming a complete block, the length of the last but one will be bigger than BlockSize. As a result, the sizes of the data blocks are different. The size of other blocks is all BlockSize except the end block.

25

After calculating the number of blocks (nBlockCnt), $EndBlockId = BeginBlockId + nBlockCnt$.

30 We can see from the description above, the length of the data block is not always unchanged. There are two types of length: one is the specified length, while the other is the remaining length of the end block.

Extract the following items of this version data specific to each version data:

- a) Attribute and identity mapping information of the overall version data;
- b) Attribute and identity mapping information of all files;
- c) Identity mapping information of each data block;

5 Make the corresponding file according to the extracted information. Refer to Figure 1 for the index file. The content is as follows:

- a) The attribute and identity mapping information of the overall version data includes:

10

PkgId	Type identity of the version data. Different types of version data adopt different PkgId. For different games, the PkgId of the version data is different;
BlockSize	Unit size of the file block (unit byte), i.e. the length of the data block
PkgIdxVer	Index version, identity of version data of the same type but different version data. For example, a same game has different version, and then PkgId of different version is the same, but PkgIdxVer is different.
SelfCRC	The self-check of CRC in the index file is used to examine whether the index file itself is correctly transmitted during the network transmission process;
FileDescList	List for the file to describe the information (several files describe the array of the information). It shows the attribute information and identity mapping information of all the files in this version data.

- b) FileDescList, the list for the file to describe the information, includes the attribute and identity mapping information of all files in this version data. As far as each file is concerned, the description information (FileDescInfo) is as follows:

15

FileAttr	File type, including directory and file;
FilePathName	Relative path included in the file name;
FileModifyTime	Time for file modification;
FileCRC	CRC check code of the whole content of the file;

FileLength	File length (unit byte);
BeginBlockId	Number of the first data in the file; as far as the entire version data is concerned, the numbers of all the data blocks are unique. Through the number of data block, the position of the data block in the entire version data and the whole file can be located quickly;
BlockDescList	List of file data block information (array of several data block information). It shows the identity mapping information of all data blocks in this file;

c) BlockDescList, the file data block information list, includes the identity mapping information of all data blocks in this file. As far as the data block is concerned, its description information (BlockDescInfo) is as follows:

5

BlockCRC CRC check code of the content of the data block.

Later, transmit the version data and index file to the synchronization source node (SyncSrcNode) to facilitate BarServer to incrementally updating the game data.

10

2. Synchronization source node (SyncSrcNode): the function of this program is to accept the index file submitted by PMUpdate program, and generate demand data block (data block that requires updating) through difference comparison. It claims corresponding data blocks from PMUpdate program, and preciously put these data blocks to the target positions according to the information inside the index. The data stored in the program are all data files with compilation. Meanwhile, SyncSrcNode also provides the synchronization services of SyncSrcNode data and BarServer of games without their versions updated.

15

20 Refer to Figure 2, and the steps of difference comparison are as follows:

a) Find out the excluded files in previous version data and all changed files according to the attribute and identity mapping information of all files in the later version index file;

25 As for the excluded files in the previous version data, update directly. In other words,

copy this file in the later version data to the previous version data;

As for the changed files, find out changed data block in the previous and later version data according to the identity mapping information of each data block in the file, and update the changed data blocks. In other words, copy this data block in the later version data to the previous version data;

b) Find out excluded files in the later version data and changed files according to the attribute and identity mapping information of all files in the previous version index files;

As for the excluded files in the later version data, delete this file directly;

As for the changed file, find out excluded data blocks in the later version data according to identity mapping information of each data block in the file, and delete this data block directly.

3. Global synchronization source server program (GlobalSSS-GlobalSyncSrcServer): its function is to distribute the resource information of the newly added game. For example, assign game ID to the newly added games and assign version number to the updated games. Also, it records the attribute of the latest game version and provides to the Internet bar server end for display. Another important task carried out by this program is to manage the game data synchronization of SyncSrcNode to check which game versions of synchronization SyncSrcNode are connected and downloaded for SyncSrcNode or BarServer. Since this program is responsible for the distribution of game resource attribute, the database shall be supplemented for persistent game information.

4. Local synchronization source server program (LocalSSS-LocalSyncSrcServer). This program itself is an agent program. It provides index files required for the game updating by BarServer as well as information concerning the SyncSrcServer node of all latest game versions at IDC end.

PMUpdate is the initiator for game updating, which provides index files and game

data after compilation. SyncSrcNode is the maintainer and provider of game data. In fact, BarServer is also a SyncSrcNode, but it also has other functions. In terms of game updating, BarServer can be comprehended as a SyncSrcNode with relatively low level.

5

GlobalSSS has two major functions, namely, the distribution of game ID and version, persistence of game attribute information as well as the synchronization of game information of SyncSrcNode.

- 10 Originally, there was only GlobalSSS without LocalSSS. Due to the order of magnitude of BarServer and with the purpose of assuring GlobalSSS free from outer attack, LocalSSS thus emerges. LocalSSS is the function cutting version of GlobalSSS. It provides the latest index file, holds the SyncSrcNode list of this version of game, and realizes the load balance effect.

15

Refer to Figure 5 to see the interaction between the updating flow and program between IDC layers:

- 20 After PMUpdate successfully logs in GlobalSSS, it requests GlobalSSS for the game list. When the game making personnel discover that there is no game to update at IDC end, they decide to add this game. Through interface interaction, PMUpdate uploads the game information edited by the game making personnel to GlobalSSS with the game ID of 0, showing the game ID is not yet assigned.

- 25 Upon receiving the game attribute information, GlobalSSS finds that the game ID is zero, and it will automatically assign a new game ID to PMUpdate program and also insert a record to the data-specific table. This record shows the attribute of this game.

- 30 After receiving the game ID number and the initial version number of GlobalSSS, PMUpdate begins to carry out compilation on the game data and starts to make index files, and also send the completed index files to SyncSrcNode of the program configuration. When the object SyncSrcNode receives the index file, create object file (filed in data compilation format) under the directory, and search the local index file that corresponds to the game ID. Then, carry out difference comparison. In the

operation of the newly added game, since SyncSrcNode local has no previous version, the difference comparison result requires the updating of all data. Then, SyncSrcNode will claim corresponding data blocks from PMUpdate program according to the demand data block list compared. After receiving block request
 5 signaling, PMUpdate reads corresponding data blocks from cache or disk and transmits them to SyncSrcNode program. After receiving the data blocks; SyncSrcNode writes into the specified position. SyncSrcNode informs PMUpdate that the updating is finished until all the data blocks are put into right positions. PMUpdate submits the index file to GlobalSSS which will locally persist to this
 10 index file, and modify relevant information of the index file to that record just inserted.

When the abovementioned operations are finished, GlobalSSS informs all LocalSSS and SyncSrcNode registered in it to request game downloading information from it
 15 (including the latest index and SyncSrcNode list for downloading). If SyncSrcNode program finds the index is different from local, it will propose the request for downloading this version of game from the SyncSrcNode program in the SyncSrcNode list obtained to start the downloading process (since this process is similar to the requesting downloading process of BarServer, it will be introduced
 20 below). After receiving downloading list, LocalSSS will inform all BarServers that are downloading the last version of this game to download the new version from a certain SyncSrcNode node.

Meanwhile, SyncSrcNode of the latest game version reports its information to
 25 GlobalSSS which will then distribute it to all registered LocalSSS.

The process for PMUpdate to upgrade a certain existing game is similar to the foregoing. The difference is that GlobalSSS does not reply game ID but the new version number.

30

Refer to Figure 6. The data updating processes and the program interactions of IDC layer and Internet bar server layer are as follows:

After BarServer successfully logs in LocalSSS, it will obtain the game list regularly.

When some games are set as automatic updating (i.e. the function of automatic game synchronization without manual intervention when the IDC is updated) or manually intervened by the webmaster, BarServer will request LocalSSS for the downloading information of this version of game. After receiving this information, BarServer will
 5 compare the new and old indexes to carry out different comparison and generate demand data block list. Claim SyncSrcNode in the downloading information for demand data block. When downloading and assembling all the data blocks, BarServer will inform the updating information of its game version of LocalSSS connected.

10

As a matter of fact, the synchronization between SyncSrcNodes of IDC layer is extremely similar to this process. The only difference is that BarServer can be viewed as SyncSrcNode for synchronization request. GlobalSSS is equal to LocalSSS here.

15

Refer to Figure 7. The game updating processed and interactions between programs in Internet bar server layer and Internet bar computers are as follows:

When BarClient is on line, it will log in BarServer. When BarServer is successfully
 20 authenticated, BarClient requests BarServer for game list. After receiving the game list held by BarServer, BarClient determines which games need downloading according to its own mechanism. It will request the index file for games requiring downloading. After receiving the index request of a certain BarClient, BarServer will send the corresponding game index back to BarClient. BarClient will work out the
 25 demand data block according to difference comparison algorithm and request demand block from BarServer. BarServer replies the content of the specified data block according to the request. BarClient obtains demand data block to determine whether a certain file is complete. If the file is complete, rename this file from temporary file to formal file.

30

After all game data files are updated, they are exactly the same as the formats and contents before the making of PMUpdate.

We can see from the foregoing that no matter it is SyncSrcNode, BarServer or

BarClient, the data updating method based on fragment index and supporting difference comparison of this invention has been used in terms of incremental updating function. The data updating method of this invention runs through the entire three layers of updating mechanisms.

5

Furthermore, as for the applications of the synchronized updating of BarServer or other SyncSrcNodes, SyncSrcNode will open relevant package files and make replies according to the application submitted. When PMUpdate uploads a game to SyncSrcNode, a lot of BarServer or SyncSrcNode will download this game and send
10 the same application, thus causing the repeated reading of the same data block. As a result, the efficiency becomes very low, and it is hard to improve the transmission speed. Then, cache the blocks that are read, and make direct replies when there are same requests.

15 Refer to Figure 8. In the three-layer architecture of this invention, the optimal downloading cache method has been adopted during the data interaction, especially during the transmission of the data block in the two layers where data interaction is carried out in order to improve the throughput and transmission speed of the data volume. The method is as follows:

20

(1) Create a shared cache (DownCache) at the supplier end of the data block. When updating the data block, the supplier end of data block will read the specified data block to the shared cache according to the request of the request end of the data block. Create the shared cache to realize uniform management of all the requests of
25 the client (request end of the data block).

As for the three-layer architecture adopted in this invention, the supplier end of the data block and the request end of the data block are relative. In the adjacent two layers, the high-ranking layer can serve as the supplier end of the data block, while
30 the low-ranking layer can serve the request end of the data block.

(2) When the supplier end of the data block receives the downloading application of the data block, first read the shared cache. If this data block exists in the shared

cache, transmit the requested data block to the request end of the data block immediately;

5 If this data block does not exist in the shared cache, submit the downloading application of this data block to the Request Queue instead of reading the file at once.

(3) When the supplier end of the data block is idle, process the downloading application in the queue according to the time of submission and transmits the requested data block to the request end of the data block.

10

As for preference, sort the data blocks in the shared cache according to the hit rate (the hit rate is the number of times for reading the same block). When the shared cache is full, the data block at the end will be eliminated from the shared cache.

15 In order to avoid that the downloading application of a certain data block waits too long in the request queue, the threshold of the waiting time is set. When the waiting time of the downloading application of a certain data block in the request queue exceeds the threshold value, the downloading application of this data block will be processed with priority even if the supplier end of the data block is not idle.

20 Generally speaking, the time threshold can be set to 15 to 30 minutes.

In order to further reduce the time for track seeking when reading and writing the file, carry out ascending sort according to the serial number for the data blocks with the same hit rate in the shared cache.

25

When updating the data block, this invention has adopted the method of optimal downloading cache to deal with massive requests in terms of the transmission of the data block. The supplier end of the data block has followed a certain strategy, i.e. give priority to the loading of games with many connections. Also, the ascending

30 sort has been carried out according to the serial numbers of the file and the data file so as to further reduce the trace seeking time when writing files and make sure the writing of files is according to the sequence.

Since the bandwidth of IDC layers is a rare resource with relatively high cost, the

IDC updating amount of this invention is already saturated at the peak moment. This invention also adopts P2P technology in order to guarantee the updating demand of the server layer of cybercafé, fully utilize the upstream bandwidth resources of the cybercafé, and allow the servers in the Internet bar to share data with each other.

5

TCPP2P uses TCP/IP protocol to carry out P2P technology with data sharing. The advantage is that TCP/IP protocol can provide stable and reliable connection to guarantee the high speed and data correction of network transmission. The establishment of connection is subject to influences from many aspects. Therefore, it is required that the Internet bar server using TCPP2P must support UPNP technology. TCPP2P needs to go through two stages: 1. Information sharing stage; 2. Data interaction stage.

Information sharing stage:

15 Refer to Figure 9. In this stage, each server in the Internet bar is required to report the game information existing in it to LocalSSS and also the information concerning IP and port supporting P2P for the connection of servers from other cybercafés. In the meantime, when the server needs to update, it can claim and report the P2P information at the server end of the cybercafés concerning this version of game information from LocalSSS connected.

In order to realize the sharing of P2P information, a class of programs called Global P2P Information Redirector (GlobalP2P) has been newly added on the basis of maintaining the original architecture of the game updating channel. Its function is to forward the P2P information reported by the registered LocalSSS to all other registered LocalSSS except the reporter. When BarServer obtains relevant P2P information, it will automatically connect to the Internet bar server end with object resources.

30 Data interaction stage:

We have introduced that BarServer has the function of SyncSrcNode. The process for the data exchange between two BarServers is in line with the processing method for the game updating process introduced before.

This invention updates the game updating to the final updating of the Internet bar from IDC layer to terminal layer through the establishment of three-layer updating method (the computer of the Internet bar is the terminal layer, the server of the Internet bar is the bar layer, and the Internet server is IDC layer). Through the three-layer updating method, the updating volume at the terminal is collected to the bar layer so as to effectively reduce the updating volume and solve the problem of the concurrency choke point of abundant terminals and a huge updating volume.

Through the segment index and difference comparison of the content of the game as well as P2P distribution network technology, this invention control the updating volume of the game to the minimum. Also, under the condition of a limited Internet server bandwidth, this invention has fully utilized the upstream bandwidth resources of the Internet bar, and shortened the time required for game updating, thus providing more carefree entertaining experience for the Internet bar and the netizens.

Claims

1. A data updating method based on data block comparison has its features lie in the following steps:
 - 5 generating a plurality of version data;
dividing the version data into several data blocks, that specific version data is extracted from each of the version data according to
 - a) Attribute and identity mapping information for the overall version data;
 - b) Attribute and identity mapping information for all files; and
 - 10 c) Identity mapping information of each data block;
 creating corresponding index files specifically to each version data, the index file includes the information extracted from the version data;
 finding out excluded files in previous version data and all changed files according to the attribute and identity mapping information of all files in the later version index
 15 file, whereby the excluded files in the previous version data are update directly and the changed files are updated upon finding out the changed data block in the previous and later version data according to the identity mapping information of each data block in the file; and
 finding out excluded files in the later version data and changed files according to the
 20 attribute and identity mapping information of all files in the previous version index files, whereby the excluded files in the later version data are deleted directly and the changed file are deleted upon finding out the excluded data blocks in the later version data according to identity mapping information of each data block in the file;
 characterized in that upon dividing a certain version data into several data blocks, a
 25 single file is taken as an object and the files are divided one by one according to the length of a pre-designated data block, that an end block is created by setting up threshold values; working out a ratio between the length of the end block and the length of the pre-designated data block to calculate the length of the end block; and combining the end block to the previous data block provided that ratio is smaller
 30 than threshold values or dividing the end block to a data block separately provided that the ratio is greater than or equal to the threshold values.

2. A data updating method according to Claim 1 further comprising a supplier end of the data block being stored in the later version data and a request end of the data block being stored in the previous version data, upon updating the data block, wherein

- (1) A shared cache has been created at the supplier end of data block. When updating the data block, the supplier end of the data read the specified data block into the shared cache according to the request of the request end of the data block;
- (2) When the supplier end of the data block receives the downloading application of the data block, first read the shared cache. If this data block exists in the shared cache, transmit the requested data block to the request end of the data block immediately;
- If this data block does not exist in the shared cache, submit the downloading application of this data block to the Request Queue;
- (3) When the supplier end of the data block is idle, process the downloading application in the queue according to the time of submission and transmits the requested data block to the request end of the data block.

3. A data updating method according to Claim 3, wherein the data blocks in the shared cache are sorted according to the hit rate and cleared up from the shared cache when the shared cache is full.

4. A data updating method according to Claim 4, wherein the data blocks are sorted in an ascending manner with same hit rate according to serial number.

5. A data updating method according to Claim 3, wherein the supplier end of the data block processes the downloading application of this data block with priority upon a waiting time threshold is exceeded.