БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ • BULGARIAN ACADEMY OF SCIENCES

ПРОБЛЕМИ НА ТЕХНИЧЕСКАТА КИБЕРНЕТИКА И РОБОТИКАТА, **58**
PROBLEMS OF ENGINEERING CYBERNETICS AND ROBOTICS, **58**

София • 2007 • Sofia

SAN Storage Provisioning with "External" Load Balancing Mechanism

Krasimir Miloshev

EMC Corp. USA, 220 Archer Ct, Malvern, PA 19355
E-mails: kmiloshev@netzero.com   Miloshev_Krasimir@emc.com

1. Introduction

The Storage Load Balancing is just one part of the general load balancing problem in computer systems. We have storage controllers/processors, which act simultaneously and number of storage units which are known as volumes connected to those storage processors. The goal is to distribute those volumes, which also are called LUNs, among the storage processors in order to achieve better performance. LUNs actually are result of mapping previously created storage devices with the RAID Storage Systems to the computer SCSI devices in order to get these devices visible from host perspective. Our focus in this article would be only the FC-based (fibre channel based) SAN storage systems.

The Storage Load Balancing is not only about the partitioning of the capacity, it's mostly about the partitioning of the performance. In SAN we have different type of RAID groups – mirrored groups (RAID1, RAID 1/0) and parity groups (RAID3, RAID5). They have different performance. So obviously the capacity spreading is not the only factor contributing to the load balance in SAN, but the storage groups RAID type as well.

The Storage Capacity Planning is first and most important step to undertake when we build our storage environment. When we do capacity planning we have three basic steps to undertake.

The entire process of planning and serving the SAN storage is called Storage Administration. Storage Administration consists of two phases: storage capacity planning and storage provisioning.

1.1. Storage capacity planning:

In order to do precise capacity planning, we have to figure out couple of very important parameters such as I/O load, what is expected to be disks load, and finally-how many disks of certain type we would need.

1.1.1. Determining the required I/O load

We have to speak with the client. He knows what applications will run, what applications upgrade he needs to implement, what is the total storage to be used, what is the percentage of Read operations and what is the percentage of Write operations (a.k.a Reads and Writes), and what is the total Host IOs. What type of RAID he wants (for instance RAID 10, RAID 5) based on his financial frame.

1.1.2. Determining disks load

We have Read and Write operations in the storage array. Writes consume more resources then Reads due to some protection mechanisms. Each manufacturer provides Best Practices Manual Book recommending how to figure out disk load. Here is some data from the EMC CLARiiON Best Practices for FC Storage manual:

Table 1

Parity RAID device	Disk IOPs= Read IOPs + 4 × Write IOPs
Mirrored RAID device	Disk IOPs= Read IOPs + 2 × Write IOPs

For instance if the client wants RAID-5 and has 20,000 Host IOs and 70% of them are Reads and 30% are Writes, then we will get 0.7×20,000 + 4×0.3× 20,000=38,000 disk IOs.

1.1.3. Calculating needed capacity

Calculating capacity means to determine the disk drives required. We are suppose to divide the total IOs by per disk IOs. This is the approximate number of drives you need to service the proposed I/O load. For RAID-5 load of 38,000 disk IOs we will need 38,000/180=212 disk drives. If RAID 10 is considered, then we will need 28,000/180= 156 disk drives. From a cost-for-performance perspective RAID 10 is a better choice in this case.

1.2. Storage provisioning

Once the capacity planning is done and certain storage capacity is provided, the next step for the storage experts is to optimize the usage of that capacity in order to keep best performance in our SAN. The usage of that capacity is implemented via storage provisioning requests. Optimizing storage provisioning tasks with regards to the best SAN provisioning practices means to distribute the requested storage load the best possible way from performance perspectives. One of the best known approaches would be searching for optimal load balance between the storage processors.

2. Storage provisioning with balancing the load in SAN

One of the most important tasks performed by the Storage Administrators is the storage provisioning. They receive requests for creating certain number of LUNs with certain size and certain RAID configuration. And the next step would be to present those LUNs to specific hosts and create volumes on those LUNs. Once volumes are created, the next step with be to create or expand file systems. But the initial step in all this process is creating the LUNs.

Most of the current storage systems consist of specific number of disks and specific number of storage controllers or storage processors. For instance in a storage system consisting of two storage processors-SP1 and SP2 we can create RAID groups over those disks. And once the RAID groups are created we can start creating/binding LUNs with specific size. That task is called storage provisioning from the LUNs point of view. Creating volumes and file systems is the last step-it's called storage provisioning from the Volume Manager perspective.

From the experience with different storage systems (EMC CLARiiON, ESS Shark) perspective we can conclude that practically there is no any efficient load balancing mechanism when Storage Provisioning operations take place. Of course, there is an internal load balancing mechanism for each system such as the "Auto Assign" option in CLARiiONs, but according to the "Best Practices Recommendations" this option should be excluded when a path failover software is used. Since most of the companies have path failover software installed, this internal option is practically very restricted. We have similar situations with some other systems such as ESS Shark, etc.

When using failover software, the "Best Practices Recommendations " require to disable the auto LUN assignment mechanism, which once again, is the internal built-in load balancing mechanism. In this situation, the failover software, not auto assign, controls ownership of the LUN in a storage system with two SPs.

Also generally with regards to load balancing the Storage Best Practices are to create even numbered LUNs and assign to SP1 and odd numbered LUNs and assign to SP2, if there are two SPs.

It is also not a good idea to assign all of the LUNs in a Raid Group to the same SP.

Also most of the path failover software packages (such as EMC PowerPath, Veritas DMP etc.) do have load balancing options as well, but these are hosts load balancing features only. In other words – when we do storage provisioning and create LUNs, we have to distribute those LUNs among the both SPA and SPB "manually". Which LUN goes where-that is the load balancing task. Our goal of course is to distribute these LUNs the best possible way, and the size sum of all LUNs connected to SP1 to be equal or close to the size sum of all LUNs related to SP2.

Let us have a practical example, using one of the most popular storage arrays-EMC CLARiiON.

From architectural point of view each CLARiiON contains so called disk enclosures with certain number of disks mounted on each disclosure.

As we already mentioned, al load in FC SAN means not only capacity, but mostly performance, because we have different type of RAID groups (RAID1, RAID1/0, RAID5, etc.). But very often we have very homogenous SAN environment – for instance only FC SAN – based CLARiiONs and only RAID5 devices. In such cases we can consider only the capacity as a factor for achieving balanced load.

Also from disk drive type perspective in CLARiiON we have: 1) FC drives (73 GB, 146 GB, 300 GB); 2) ATA drives (320 GB) and 3) SATA drives (250 GB). To simplify the research process we also will consider in this article only FC drives.

Lets take a look at one possible CX700 example: The EMC CE (custom engineer) has created first 20 RAID5 raid groups using disks $0,...,6$ from each disclosure, then another 20 RAID5 groups using disks $7,\ldots,13$ from each disclosure and finally 20 hot spare raid groups using disk 14 from each disclosure.

And the next step is to run so called "LUN binding" in order to create LUN with specific size and assign it to ether SPA or SPB. It's very important to keep relatively equal load over the two storage processors SPA and SPB in order to achieve higher performance.

Generally speaking there are two possible approaches for load balancing in SAN Storage Systems – static and dynamic provisioning.

2.1. Static provisioning with balancing the storage load

According to this approach the Storage Administrators preliminary create LUNs with specific size and distribute those LUNs over the two SPs–SPA and SPB. This could be quite long manual process. They don't have much flexibility and usually pick two or three LUN sizes (for instance 25 GB and 50 GB LUNs). Then they can create metaLUNs to get bigger volumes when this is necessary. The normal practice is to create only striped metaLUNs across different RAID groups.

Obviously one of main disadvantages here is the lack of flexibility. If we have to type of LUNs with regards to the size-for instance 25 GB and 50 GB, we don't have so many options when we crate metaLUNs. If somebody needs 60 GB LUN and company has the practice of usage only striped metaLUNs (which of course is the right approach in order to get better performance), we have to offer him metaLUN made up of 3×25 GB LUNs. Thus we just wasting a disk storage. If someone needs 160 GB LUN, we have to create metaLUNs made up of 4×50 GB LUNs. In this case we waste 40 GB disk space.

2.2. Dynamic provisioning with balancing the storage load

When we do storage provisioning (which means creating certain number of LUNs, each of them with some specific size), we don't create LUNs in advance. We don't distribute LUNs among the two SPs in advance. We don't waste disk space. We don't look at any preliminary created data spread sheet for storage allocation. We just create LUNs with the requested sizes and spread these LUNs among both SP1 and SP2 based on load balancing algorithm, which can be implemented easily in many ways (via Korn-shell scripts for instance, or via Perl scripts). Every time when we have storage provisioning task (to allocate certain number of LUNs with certain sizes) we create these LUNs and then just run this script and get load balanced distribution among the two processors SPA and SPB. Thus we eliminate necessity of keeping and maintaining long records of what has been used last and avoid any subjectivity. Also we eliminate necessity of creating and distributing LUNs in advance. In this case we will create metaLUNs only when our client requires volume expansion. With so called static approach we totally depend on the "Last RAID group used" and "Next RIAD Group to be used" records. Also if somebody makes mistake and misses something, the whole approach falls down. Every time when we do storage provisioning and have to create large number of LUNs, we run the load balancing script, which would give us the SP location (ether SPA or SPB) of each of the LUNs to be created. And then we create and assign to the SP all the LUNs manually in accordance with the script proposal.

In order to simplify furthermore the task we would consider that we have only one type of RAID devices (for instance RAID5 LUNs and no meta-LUNs).

Let us have a Storage Provisioning request for creating 8 LUNs (Table 2).

Table 2

LUN No	Size in GB
LUN[1]	25
LUN[2]	30
LUN[3]	10
LUN[4]	60
LUN[5]	45
LUN[6]	150
LUN[7]	70
LUN[8]	50

We can sort these LUNs by size (in GB) and then we will get the values shown in Table 3.

Table 3

S[1]	S[2]	S[3]	S[4]	S[5]	S[6]	S[7]	S[8]
150	70	60	50	45	30	25	10

Then the next step is to distribute over the two SPs those eight LUNs the best possible way from load balancing perspective.

The total capacity sum of all eight LUNs= 440. That means the ideal case would be to distribute those LUNs among the two SP in a way the capacity related to each SP to be 220 GB.

Let us use one particular load distribution algorithm, proposed in our discussions by Prof. St. Stoichev , which is based on using two indices *i* and *j*.

The index "*i*" increases from bottom up, and "*j*" decreases from top down. If we have a total capacity of 440, then we can introduce so called pivot, which is half of that value.

1. S[1] goes to SPA and S[2] goes to SPB.
2. For S[8] if S[1] + S[8] < 220, then S[8] goes to SPA and S[7] goes to SPB.

Otherwise S[8] goes to SPB, not to SPA.

3. For S[3] if S[1] + S[8] + S[3] < 220, then S[3] goes to SPA and S[4] goes to SPB.
Otherwise S[3] goes to SPB, not to SPA.

4. For S[5] if S[1]+ S[8] + S[3] +S[5] < 220, S[5} goes to SPA and S[6] goes to SPB.
Otherwise S[5] goes to SPB, not to SPA.
The final distribution is indicated in Table 4.

Table 4

SPA	SPB
S[1]=150	S[2]=70
S[8]=10	S[7]=25
S[3]=60	S[4]=50
	S[6]=30
S[1]+S[8]+S[3]=220	S[2]+S[7}+S[4]+S[6]=220

We can see that we have in this case the perfect load balancing scheme – 220 GB to each SP.

3. Program implementation

We can easily implement the load balancing algorithm by using C or Korn-shell for instance.

1. The first step would be to sort by size all those LUNs requested for storage provisioning purposes. We can use so called bubble sort algorithm to achieve that.
2. And the next general step would be to follow up the load distribution algorithm discussed already in the chapter above.

- Here is an implementation in C for the bubble sort part of the algorithm.

```
    int i, j, temp;
  for (i = 0; i < (array_size – 1); ++i)
  {
      for (j = 0; j < array_size – 1 – i; ++j )
      {
          if (S[j] > S[j+1])
          {
              temp = S[j+1];
              S[j+1] = S[j];
              S[j] = temp;
          }
      }
  }
}
```

- Here is an implementation in Korn-shell for the distribution part of the algorithm

```
SPA=0;SPB=0;i=1;j=n;p=1;SUM=0; k=1
for p=1..n do SUM=SUM+S[p]
PIVOT=SUM/2
```

```
While i< j do
      begin
         if SP1 < PIVOT
           then
            begin
              SP1=SP1+S[i] + S[j];
                  IND1[k]=I;
                   IND[k+1]=j;
                    k=k+2;
                    SP2=SP2+S[i+1]+ S[j –1];
                   IND2[k]=i+1;
                  IND2[k+1]=j–1;
                 i=i+2;j=j–2
              end
             else
              begin
              SP2= SP2 + S[i]+S[j];
                  IND2[k]=I;
                   IND[k+1]=j;
                    k=k+2
                   SP2=SP2+S[i+1]+ S[j-1];
                   IND2[k]=i+1;
                  IND2[k+1]=j–1; i=i+2;j=j–2
              end
        end
```

4. Conclusion

In small and mid-size corporate SANs usually we don't have intensive requests for Storage Provisioning. For instance they normally request couple of LUNs to be created on daily or even on weekly basis. Obviously only in large companies with large SANs and high number of storage devices we can have high numbers of daily requests for storage provisioning and creation of LUNs. Also in such the servers are usually clustered, so the internal auto-assign option usually is turned off, especially if those storage systems are particularly CLARiiON and ESS Shark. And in such cases this load balancing script approach would be very applicable and powerful. Otherwise in small cases we can just do load balancing manually by checking the load on SPA and SPB and binding those couple of LUNs appropriately. In cases with intensive storage provisioning requests it's better to use the load balancing script. Also we already mentioned that this load balancing mechanism would work best only when we have homogenous SAN with regards to the device type (same RAID, same disk drive type – ATA, SATA, FC). We called this mechanism "external" since it's not a part of the "internal" built-in -load-balancing architecture. And this "external" load balancing mechanism is applicable on different storage systems where the "internal" mechanism has to be turned off due to different reasons such as clustering etc. The mechanism we presented has very practical meaning – it can be used and would boost the performance on platforms like IBM ESS Shark and EMC CLARiiON in cluster configurations, when their internal load balancing scheme is supposed to be switched off. Then we can run this scripts always when we do a storage provisioning. That is the major practical recommendation of this mechanism – to be used in such storage platforms when they are put under cluster configuration and because of that their internal load balancing mechanisms are switched off. It's hard to generalize and measure precisely the exact effect when we apply this mechanism on specific systems like CLARiiON and IBM Shark, but I have run certain tests comparing the speeds of some backup operations in particular storage CLARiiON CX 500 – based configurations, placed under IBM AIX cluster (HCPMA) – in cases when we have used this load balancing script we achieved between 10-15% shorter backup running time for those particular backup jobs.

Acknowledgements. I would like to express my gratitude to Prof. St. Stoichev of Technical University in Sofia, Bulgaria. His ideas and remarks were very useful and practical.

R e f e r e n c e s

1. E r d o s, P., L. B. R i c h m o n d. On Graphical Partitions: Combinatorics and Optimization Research Report COPR. University of Waterloo Publishing, Waterloo, Ontario, Canada, 1989, 89-42.
2. W a h, B., P. M e r h r a. Load Balancing: An Automated Learning Approach. O'Reilly Media, Inc., 2001.
3. B e a u c h a m p, C., J. J u d d, B. F. K u o. Building SANs. Rockland, Syngress Publishing, MA, 2001.
4. S e d g e w i c k, R. Algorithms in C. Part 5. Boston, MA, Princeton University, Addison-Wesley, 2002.
5. H i l e s, A. Business Continuity: Best Practices. Brookfield, Rothstein Ass. Inc., CT, 2004.
6. Q u i g l e y, E. UNIX Shells. Practice Hall PTR, NJ, Upper Saddle River, 1999.
7. CLARiiON CX3-80 Technical Guide (white papers). Hopkinton, MA, EMC Corp., 2006.
8. B o u r k e, T o n y. Server Load Balancing. O'Reily Media Inc., 2002.

Задача управления памяти в SAN при помощи “внешнего” механизма балансирования

Красимир Милошев

EMC Corporation USA, 220 Archer Ct., Malvern, PA 19355
E-mails: kmiloshev@netzero.com Miloshev_Krasimir@emc.com

(Р е з ю м е)

Задача балансирования нагрузки памяти (Storage Load Balancing Problem) является часть более общей задачи Load Balancing Problem, которая широко известна компьютерным специалистам. Цель работы распределение лучшим способом нагрузки (LUNs) между процессорами по отношении оптимального функционирования. Обсуждается „внешная” схема балансирования, основана на оптимальное распределение LUNs.