Midrange computing cluster architectures for data analysis in High Energy Physics

Andrey.Shevel@pnpi.spb.ru
Petersburg Nuclear Physics Institute

Roy.Lacey@Stonybrook.edu
State University of NY at Stony Brook

Abstract
Data analysis in High Energy Physics is often performed on midrange computing clusters (10-50 machines) in relatively small physics groups (3-10 physicists). Those clusters have been built from commodity equipment and they are running under one of Linux flavors. For such kind of the groups it is not easy to get financial support regularly to do upgrade of available computing cluster when physicists feel lack of resources. That gives idea to choose right cluster architecture to get maximum of cluster performance when financial support becomes available. Author plans to describe several cluster architectures and show possible drawbacks and how to avoid them.

Simplest architecture (scenario 1)

In simplest cluster architecture which has been described many times elsewhere we have one file server where all disks (**RAID** arrays) connected to and a range of compute nodes (peripheral computers) which have access to the same user directories mounted by **NFS**. Quite often the central machine is used as access server (user has to be logged to central machine). Usually on the same machine the batch system service is running. It is very simple and cheap configuration which could be implemented in couple of days or week. Let us see how it works.
In **HEP** physics analysis we are dealing with large volumes of the data. In many cases the data volume 2-20 TB are quite usual or even small one. Let us also imagine we need to analyze 10 TB of the data. Because those data are located on central machine the data need to be copied from the disks to main memory of central machine and after that have to be transmitted to compute node over **NFS**. If we submit one job for one compute node (other nodes are free) the total time required for job performance could be something like ***Tc*** (CPU time for all the data) +

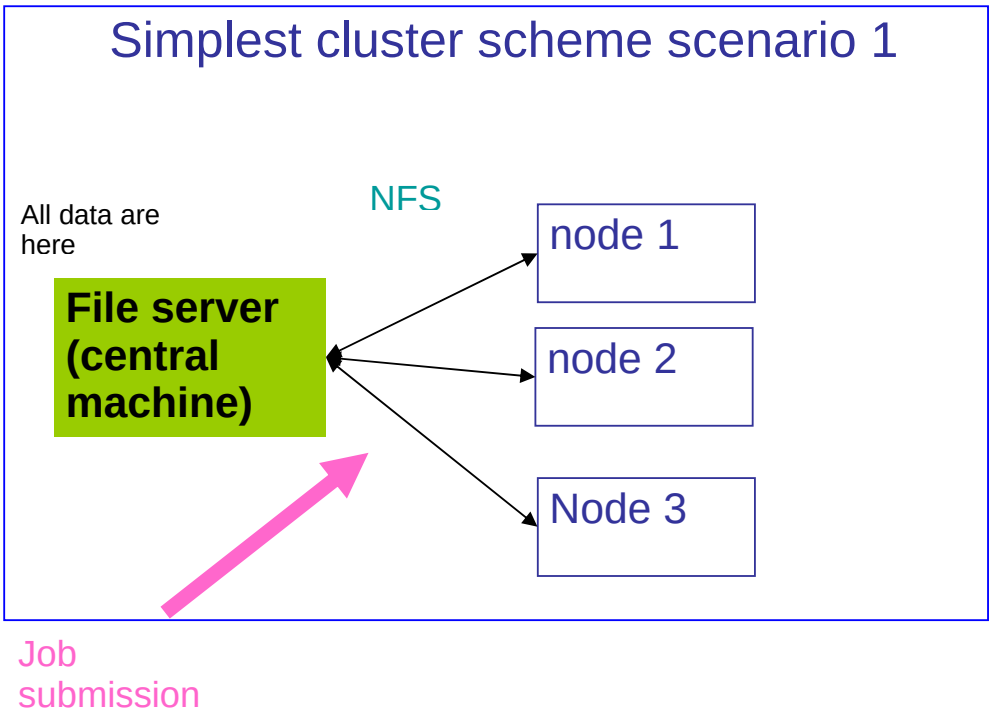

Figure 1. Case of three compute nodes.

Td (time to read all the data from disk + time to transmit all data over **NFS**). We could tell just a little about the ***Tc*** because it depends on the data itself, character of the analysis, etc.

However we could estimate the time ***Td***. In standard case when every node connected to central machine through separate Ethernet adapter connecting compute node and central machine we would have the bandwidth at about 60-80 MB/sec with 1 Gbit network and with **NFS** record size about 32KB. Raw disks + file system **XFS** could realistically deliver the bandwidth about 80 MB/sec. That means we could expect about 80 MB/sec or so. In this case the value ***Td*** is equal 10 TB/80 MB/sec = 131072 sec = 2184.5 min = 36.4 hours = 1.52 day. The calculated time is only for the data reading. As it was mentioned before we do not tell anything about CPU time. Anyway total time for the analysis will be more that 1.52 day. The situation is more unpleasant if CPU time is not large enough in comparison to the time of data transfer – CPUs will spend most time in waiting when the data are transferred.
Let us imagine physicist tries to speedup the computation. His assumption might be very simple – two or three machine definitely will be faster than one. However physicist might discover that several jobs submitted to analyze same volume of the data will require almost same time, but not twice less or three times less. It is not surprising because good part of whole process is data reading itself but not the computation.

		Time						
		1	2	3	4	5	6	7
Machine number	1	read	comp	comp	read	comp	comp	read
	2	wait	read	comp	comp	read	comp	comp
	3	wait	wait	read	comp	comp	read	comp

Figure 2. Process diagram for case of 3 compute nodes.

So we can see from above example that relation in between time for the data transfer and time for the computing (CPU time) is important. We can introduce new variable *alpha* which does mean relation in between CPU time per portion of the data (per array, or event, or other portion) and time for the data transfer (from disk to memory, from computer to computer, etc.) of same data portion. Let us discuss simple model of computing cluster. In this model we assume following limitations:

- only one compute node can read data from the file server at one time, i.e. if compute node needs data it has to wait till other compute node accomplishes reading;
- only one job can be performed on the compute node.

		Time						
		1	2	3	4	5	6	7
Machine number	1	read	comp	comp	wait	read	comp	comp
	2	wait	read	comp	comp	wait	read	comp
	3	wait	wait	read	comp	comp	wait	read
	4	wait	wait	wait	read	comp	comp	wait

Figure 3. Process diagram for case of 4 compute nodes.

The fig. 2 shows the situation when relation in between reading time and computing time is 2 (*alpha* = 2) for three compute nodes. The colors of the cells in the picture means: red – waiting for the data, yellow – reading the data, green – computation. The fig. 3 represents case of 4 compute nodes. It is interesting to compare figs 2 and 3. We see that in both cases only five full cycles have been accomplished (read-comp-comp) in the same time in spite of different number

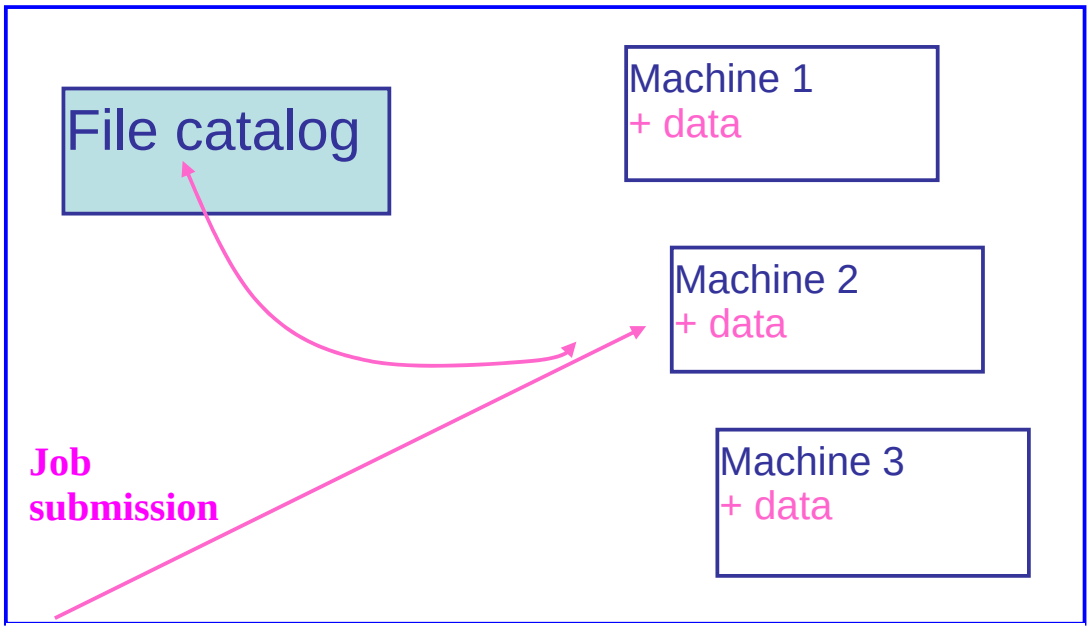

Figure 4. Compute nodes with large disk space.

of the machines. It is quite clear because part of the machines just waiting for the data from file server. Because the bottleneck is file server we can estimate what is maximum number of the machines will be effective (when all machines are loaded on reasonable level and loaded evenly) for concrete relation in between time for computing and time for reading. We could introduce the load of file server from the compute node $l=Tr/(Tr+Tc)$, where ***Tr*** – time reading of one portion of the data. Apparently the load from n compute nodes will be $n*Tr/(Tr+Tc)$, where ***n*** – number of compute nodes. Obviously the file server load could not be more than 1. So we can write down $n*Tr/(Tr+Tc) <= 1$. That means

$$n <= (Tr+Tc)/Tr = 1 + Tc/Tr$$

In our case (when $Tc/Tr = 2$) the effective number of the machines is (in according to the formula) equal to 3.

In simplest cluster scheme shown above we got simple formula which might help us to estimate how effective could be using the large number of the machines. We also can pay attention that if we upgrade the machines and do not touch other cluster components we change the relation ***Tc/Tr***. As the result the number of running jobs may be decreased. In other words you can buy less modern machines than number of old machines.

Modified scenario (scenario 2)

To reduce the ***Tr*** (time to read the portion of the data) we could modify the simplest cluster scheme in according to picture on fig. 4. Here we distribute all data files on the compute nodes (they have to have enough disk space). After that we will form the batch of jobs and submit the jobs only to the nodes where data are. In such the way we reduce the time ***Tr*** because all reading will be done locally on compute nodes but not over **NFS** (i.e. about two times faster). Here we might expect that total time to process all the data will be decreased as $1/n$, where ***n*** is number of computing nodes.

Yet another modified architecture (scenario 3)

Let us imagine that we still need for some flexibility in data location in scope of the cluster and would not like to use file catalog as it was in scenario 2. Every compute node has significant disk space, for example several **TBs**. To avoid bottleneck in the network we could use several network switches as shown on the picture fig. 5. In this picture we see that any disk space data are available to any compute node over special network segment. Obviously in this situation we need to have the number of Ethernet adapters on every node is equal to the number of network

switches. Also we need to mount the file systems differently on different nodes. For example, the file system “disk space 1” will be mounted locally on the node 1, over network switch-2 on the node 2 and over network switch-3 on the node 3. From other hand on compute node 1 all disks are available over different channels:

- disk space 1 is mounted locally;
- disk space 2 is mounted over network switch 2;
- disk space 3 is mounted over network switch 3.

The same could be said about any compute node. In this scheme we have more freedom and can use scenario 1 and/or scenario 2.

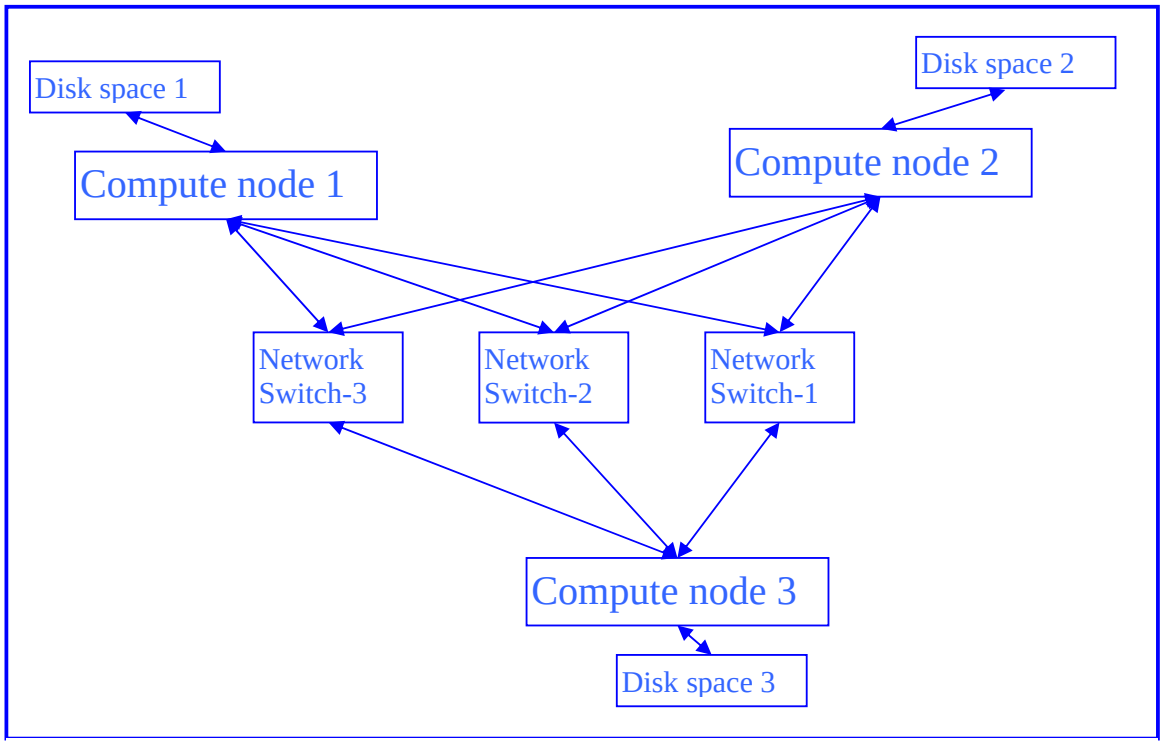

Figure 5. Modified scheme of the cluster (scenario 3).

It looks simple for three machines. For large number of machines we would need to use all available features:

- two Ethernet ports on mother board (most of server mother boards have at least two);
- two additional quadro Ethernet port cards (8 Ethernet ports in sum).

In result we will have 10 Ethernet ports per compute node. Of course we need 10 switches. In such way we can build up cluster consisting of 10 compute nodes. In such the configuration we have completely separated network traffic from one node to other node. That means no NSF interference in between different transferred data located on different nodes. In other words we have here possible data transfer throughput 10 times more than in fig.1.

In case when you need more than 10 nodes (remember we are talking about midrange clusters) it is possible to implement with the same relatively inexpensive set of equipment. For example, 48 machines. In this case you can continue to use stack of 10 network switches, 48 Ethernet ports each.

Conclusion

With permanent growth of the computing power per machine (doubled every 18 months) we need to analyze more carefully upgrade process for midrange clusters (Tier-2 or Tier-3).