

# HPSS RAIT Network Overhead Analysis

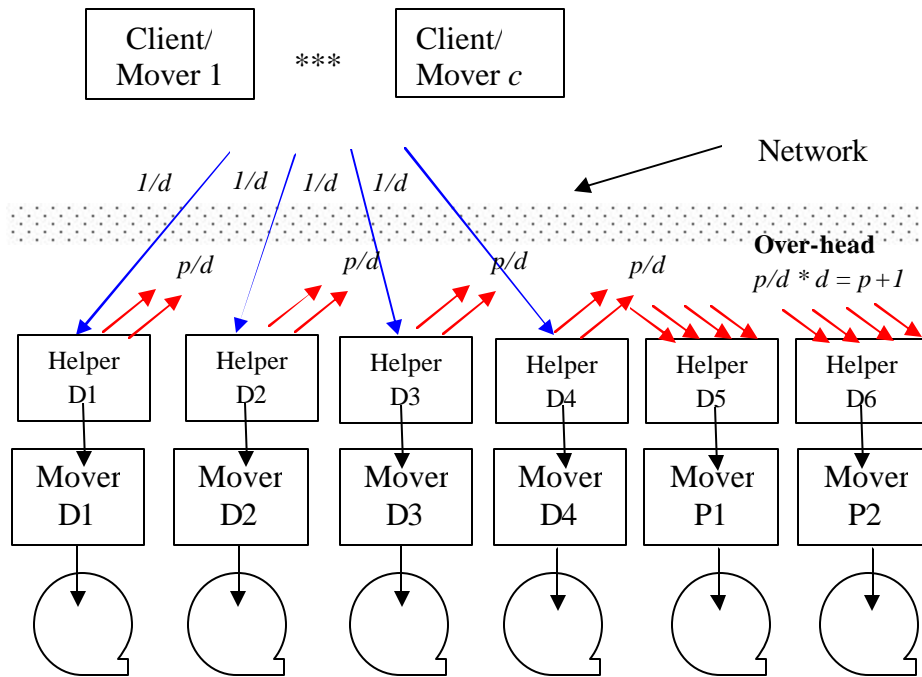
July 1, 2009

Benny Wilbanks, Kent Dehart & Jason Alt

In this discussion we assume that the HPSS Client/Mover stripe width is represented by  $c$ . The data stripe width for tape is represented by  $d$  and the number of redundancy tapes is represented by  $p$ . The total tape stripe width is  $d+p$ . In the examples we take  $d = 4$  and  $p = 2$ .

## Approach #1 – Rotating Redundancy Generation

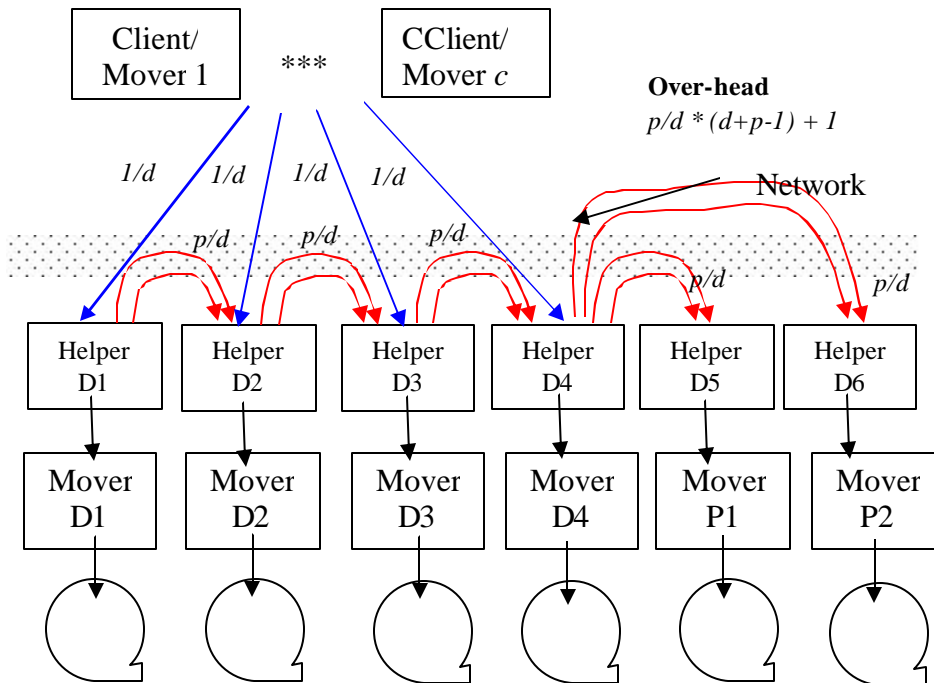
This is one of the original proposals, where  $d$  helpers (each running on a tape Mover) collects (over the network)  $1/d$  of the data and sends it to the local tape Mover, by utilizing share memory. It also transmits, over the network, this data to  $p$  helpers. These helpers will generate redundancy information and send it to the local tape Mover via shared memory. This configuration is depicted in the figure below.



Notice that the network overhead required for this approach is the transfer of  $1/d$  of the data by each  $d$  helper for each  $p$  redundancy blocks. Totaling the overhead gives us:  $(d * p * 1/d) + (d * (1/d))$  or simply  $p + 1$ . For each redundancy tape all the data must be transferred to the node targeted to generate redundancy information.

## Approach #2– Rotation Redundancy Pipeline

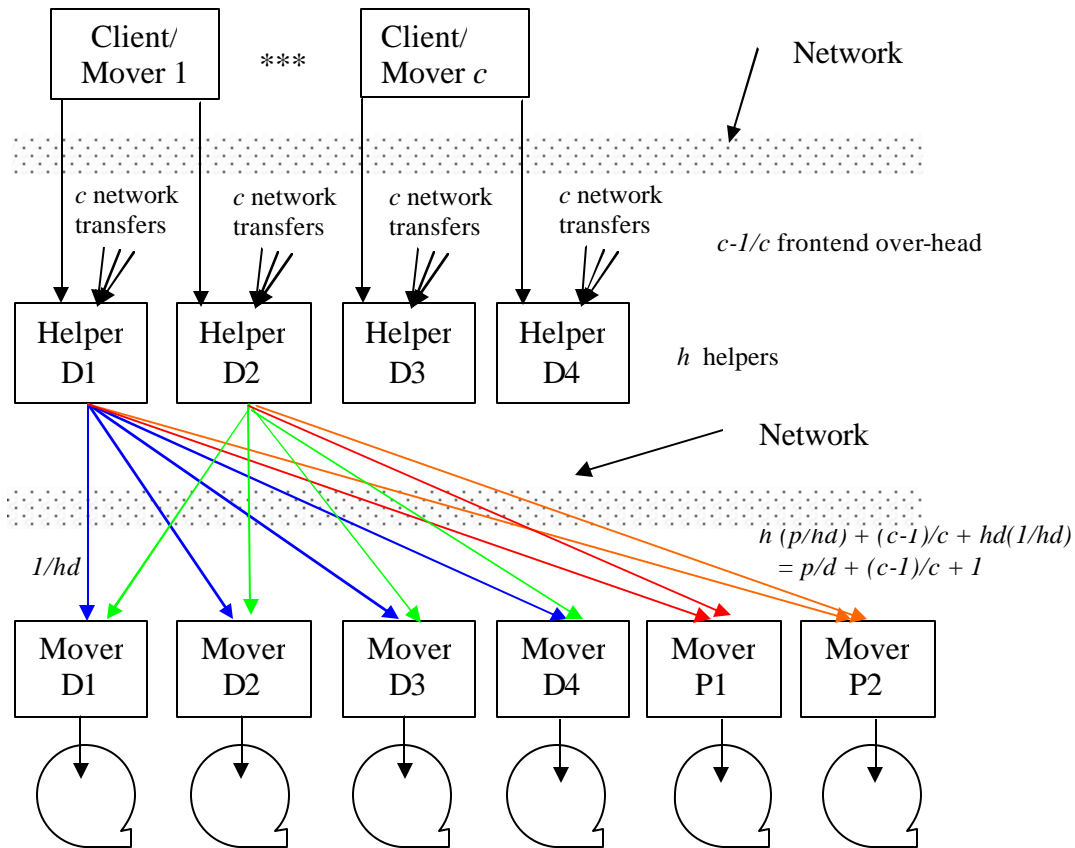
This is also one of the original proposals, where  $d$  helpers (each running on a tape Mover) collects (over the network)  $1/d$  of the data and sends it to the local tape Mover, by utilizing share memory. It also transmits, over the network, a partial redundancy block. These helpers continue receiving data, computing partial redundancy information and passing it on until the block is complete and it is passed to another helper. These  $p$  helpers will send the redundancy block to the local tape Mover via shared memory. This configuration is depicted in the figure below.



The network overhead required for this approach is the transfer of  $1/d$  of the data by each  $d-1$  helper for each  $p$  redundancy blocks. Totaling the overhead gives us:  $(p * 1/d * (d + p - 1)) + (d * 1/d)$ , or simply  $(p/d * (d+p-1)) + 1$ . This approach is more expensive than that the first. It was included for completeness.

### Approach #3– Per Stripe Redundancy

This is based on the proposal from NCSA, where redundancy information is calculated closer to the source, using the available data. Each Helper (which should be started on the Client/Mover machine) gets data for an entire stripe length extent, generates the redundancy data and transmits all to tape Movers. One or more of these Helpers can be run in parallel. In our example, we use  $h$  to represent the number of Helpers running. This configuration is depicted in the figure below.



The network overhead required for this approach is partly depended on how wide the data is distributed at the Client/Mover. Because each Helper is started on the Client/Mover machine only  $c-1/c$  network overhead is required. The remaining  $1/c$  is handled via shared memory. Additionally, since the transfer is split between the  $h$  helpers, each helper will handle  $1/h$  of the data. This includes writing  $p$  of the  $1/h$  redundancy information to the Movers. Totaling the overhead gives us:  $h(p/hd) + (c-1)/c + hd(1/hd)$ , or  $p/d + (c-1)/c + 1$ .

## Conclusion

If we concentrate on approach #1 and #3 (#2 is not even in the running), approach #3 appears to be the most efficient (see the solution of the following inequality):

$$\begin{aligned}
 & \text{Approach \#1} > \text{Approach \#3} \\
 & p + 1 > 1 + p/d + (d-1)/d \\
 & p > p/d + (d-1)/d \\
 & p - p/d > (d-1)/d \\
 & pd/d - p/d > (d-1)/d \\
 & p(d/d - 1/d) > (d-1)/d \\
 & p(d-1)/d > (d-1)/d \\
 & p > 1
 \end{aligned}$$

So Approach #3 is better than #1 when  $p > 1$ . If  $p$  is equal to 1, then they both produced the same network overhead. The included table demonstrates this.

There must be care taken with the approach because it is possible in a single Client/Mover situation to run more helpers per Client/Mover node than can be handled by the network interface. If you consider 250MB/sec tape drives,  $d=8$  and  $p=2$ , you'd need to push 3125MB/sec across your network and 2500MB/sec to the drives to keep them streaming. Also, to reach an adequate level of parallelism the Client/Mover will be required to buffer multiple HPSS stripe length of data. For example, to fully drive an 8 way tape stripe, you might have a 16 way disk virtual volume. If your stripe block size is 4MB, then each disk Mover would need to buffer 32MB of data. This is something that we need to continue to investigate for feasibility.

It may be difficult to run the RAIT helper on the same machine as the Client/Mover. The next best case may be to run it on one of the Tape Mover machines. The network overhead in this case would be:  $1 + p/d + (d-1)/d$

			Overhead Multipliers		
<i>c</i>	<i>d</i>	<i>p</i>	Approach #1	Approach #2	Approach #3
1	4	2	3	3.50	2.25
1	4	1	2	2.00	2.00
8	4	2	3	3.50	2.25
1	8	2	3	3.25	2.13
16	8	2	3	3.25	2.13
16	8	4	5	6.50	2.38
20	10	2	3	3.20	2.10
20	10	4	5	6.20	2.30
24	12	2	3	3.17	2.08
24	12	4	5	6.00	2.25

