**Evaluation for Azure-LRC**

We evaluate our repair-optimal data placement (denoted by Optimal) via numerical analysis and testbed experiments. We compare Optimal with the flat data placement (denoted by Flat) which stores each block of a stripe in a distinct cluster, and the random data placement (denoted by Random) which stores the blocks at random subject to single-cluster fault tolerance.

The overall results are summarized as follows. First, from numerical analysis, Optimal greatly reduces the cross-cluster repair traffic of Flat and Random. Second, from testbed experiments, Optimal reduces the degraded read time of Flat and Random by up to 89.1% and 87.7%, respectively, and achieves up to 3.3× and 2.7× node repair rate gains over Flat and Random, respectively.

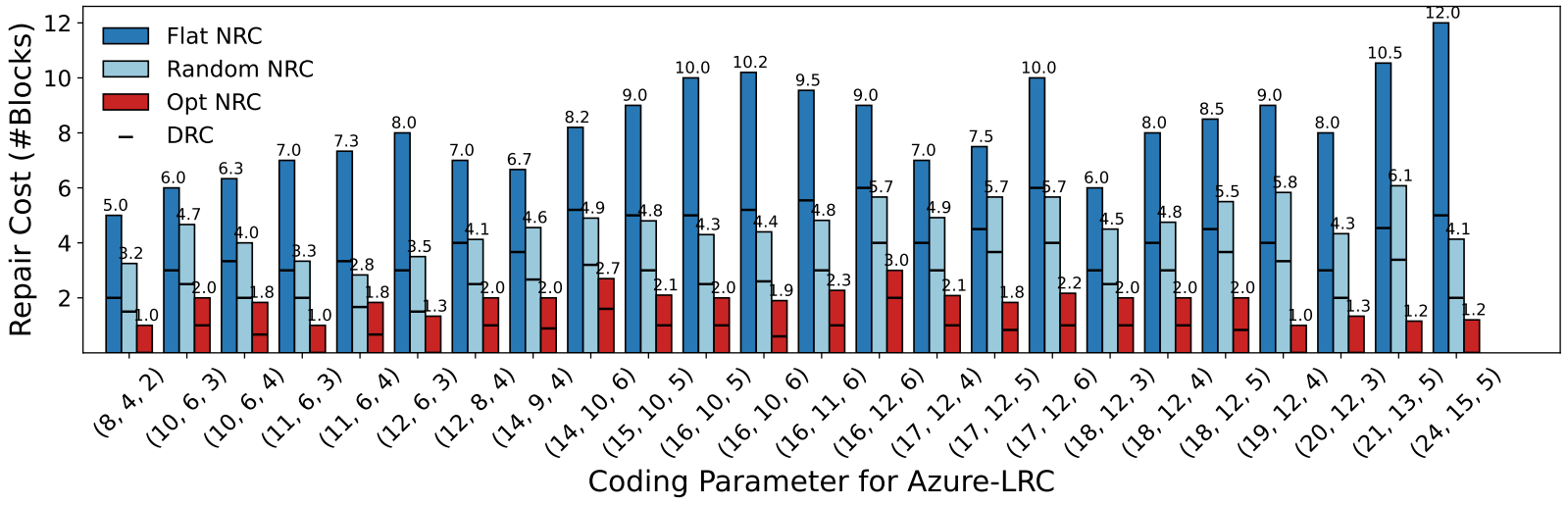
**Metrics**. We adopt the metrics defined in paper[1]. 𝐶𝑜𝑠𝑡(𝐵𝑖) denotes the number of blocks transferred across clusters to repair a block 𝐵𝑖. We adopt the two metrics to measure the repair performance of the data placement scheme for Azure-LRC. The first metric is degraded read cost (DRC), which is defined as the average number of blocks required to repair all data blocks. The second metric is normalized repair cost (NRC), which is defined as the number of blocks required to repair all blocks divided by k. Thus, DRC can be reviewed as the cost of repairing a failed data node and NRC can be reviewed as the cost of repairing a failed node. The above two metrics are comprehensive to measure the degraded read performance and the node repair performance of Azure-LRC.

**Numerical Analysis**

Comparisons of three data placements. We adopt two data placement schemes, i.e., flat data placement and random data placement, as the comparisons to measure the improvement that the optimal data placement achieved. The flat data placement is to put every block into a single cluster while the random data placement is to randomly choose random number of blocks and put them into a cluster subject to single-cluster fault tolerance.

For numerical analysis, we first consider the common parameters used in enterprise storage systems, for example (n = 16, k = 12, r = 6) (used in Microsoft Azure) and (n = 16, k = 10, r = 5) (used in Facebook HDFS). For generalization, we consider the parameters, close to those commonly used, under 8 ≤ n ≤ 24, 4 ≤ k ≤ 15, 2 ≤ r ≤ 6. And we consider the redundancy ratio(i.e. n/k) from 1.33 to 2.

The Figure followed shows the repair costs of Flat, Random, and Optimal under 24 sets of coding parameter.



We summarize the following observations.

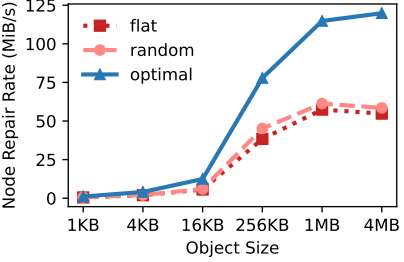
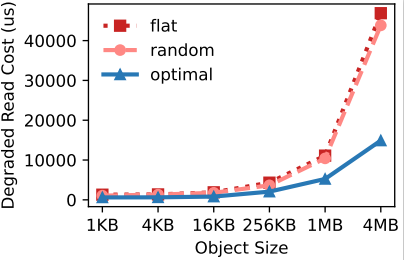
* Optimal always has the minimum NRC and DRC. Optimal reduces the NRC and DRC of Flat significantly, and also shows notable improvements over Random. For example, for (12, 6, 3) Azure-LRC, NRC = 8.0 and DRC = 3.0 in Flat, NRC = 3.5 and DRC = 1.5 in random, NRC = 1.3 and DRC = 0 in Optimal.
* For the case where d - 1 ≥ r + 1, Optimal places each local group of blocks in one cluster subject to single-cluster fault tolerance, the cost for repairing any data block or local parity block is zero but the cost for repairing any global parity block is still non-zero, implying DRC = 0 but NRC ≠ 0. For example, for Azure-LRC with parameters (8, 4, 2), (11, 6, 3), (12, 6, 3), (19, 12, 4), (20, 12, 3), (21, 13, 5), (24, 15, 5), DRC = 0.
* As r increases, NRC and DRC increase in Flat, while they keep stable in Optimal. For example, for (17, 12, 4) Azure -LRC, NRC = 7.0 and DRC = 4.0 in Flat, and NRC = 2.1 and DRC = 1.0 in Optimal, while for (17, 12, 5) Azure-LRC, NRC = 7.5 and DRC = 4.5 in Flat, and NRC = 1.8 and DRC = 0.8 in Optimal, and for (17, 12, 6) Azure-LRC, NRC = 10.0 and DRC = 6.0 in Flat, and NRC = 2.2 and DRC = 1.0 in Optimal.
* NRC is always larger than DRC because the cost for repairing the parity blocks is amortized over the data blocks.

**Testbed Experiment**

Setup. We deploy our key-value store prototype on a local cluster which consists of five physical nodes, each of which runs Ubuntu 16.04.5 LTS with a quad-core 3.40 GHz Intel Core i5-3570, 16 GB RAM, and a Seagate ST1000DM003 7200 RPM 1 TB SATA hard disk. The nodes are connected by a 10 Gbps network. We assign one node as the network core and the remaining nodes as storage servers to emulate clusters. Each cluster consists of one proxy and multiple servers. We use the Wonder Shaper tool to control the outgoing bandwidth of the network core to make the cross-cluster bandwidth constrained, emulating cross-cluster transmission.

For testbed experiment, we adopt (n, k, r) = (16, 12, 6) Azure-LRC which is adopted by Microsoft Azure. We consider various object sizes from 1 KB to 4 MB and configure the ratio of the cross-cluster bandwidth to the inner-cluster bandwidth is 1:10, i.e. the bandwidth of network core is 1Gbps. We vary some of the settings in our experiments. We measure the degraded read time, which is the average time to repair all data blocks in a stripe. We then measure the node repair rate, which is the average rate to repair all nodes in a system. Note that node repair involves the repair of data and parity blocks. The results of each experiment are averaged over ten runs.

**Experiment 1 (Repair performance under different object sizes).** We first evaluate the repair performance of Optimal under different object sizes, varied from 1 KB to 4 MB. We consider (16, 12, 6) Azure-LRC, and adopt the default bandwidth configuration, i.e. 1 Gbps of cross-cluster bandwidth and 10 Gbps of inner-cluster bandwidth. The Figures followed shows the results.

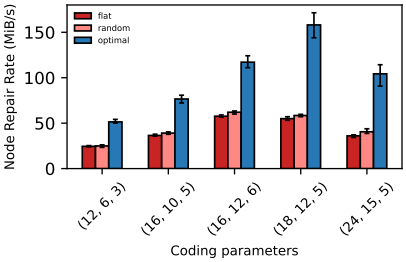
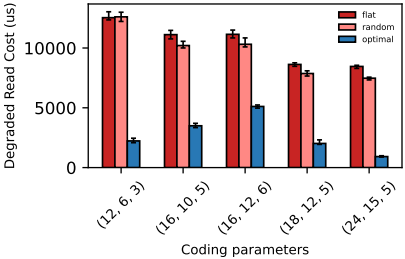


（a）Degraded Read Time （b）Node Repair Rate

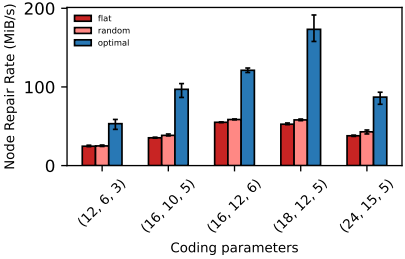
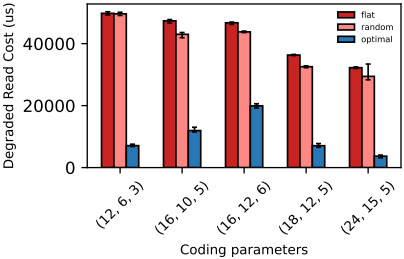
From Figure (a), the degraded read time increases with a larger object size. And we can see that Optimal constantly outperforms Flat and Random. The experimental results comply with the theoretics, Optimal has a much smaller DRC than Flat and Random. Overall, Optimal reduces the degraded read time of Flat snd Random by 53.0% and 42.6%, 53.8% and 52.1%, 56.6% and 53.7%, 52.2% and 43.5%, 52.8% and 49.6%, 68.2% and 66.0%, respectively, across all object sizes. We can see that Optimal achieves larger gains with a larger object size generally, as the cost for transferring blocks across clusters increases with a larger object size.

From Figure (b), the node repair rate also increases with a larger object size, but the line tends to flatten out or slightly decrease as the object size becomes larger when the object size touches 4Mb. A possible reason is that node repair includes parity blocks repair which greatly increases the cross-cluster traffic, leading to touch the upper bound of cross-cluster bandwidth, since the repair of a global parity block needs to retrieve the information of all the data blocks. Overall, Optimal achieves larger node repair rates and achieves 2.1× and 1.7×, 2.1× and 2.0×, 2.2× and 2.1×, 2.0× and 1.7×, 2.0× and 1.9×, 2.2× and 2.0× node repair rate gains over Flat and Random, respectively, across all object sizes.

**Experiment 2 (Repair performance under different coding parameters).** We next evaluate repair performance under different coding parameters. We consider five sets of (n, k, r), i.e., (12, 6, 3), (16, 10, 5), (16, 12, 6), (18, 12, 5), (24, 15, 5). We consider two object sizes, 1MB and 4 MB. Figure followed shows the results (the error bars show the maximum and minimum results across over ten runs).



（a）Object size 1MB （b）Object size 1MB

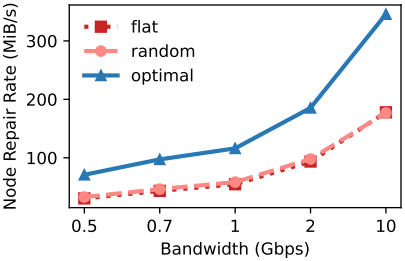
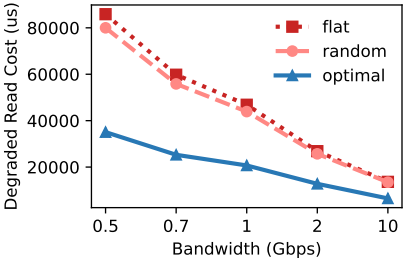


（c）Object size 4 MB （d）Object size 4 MB

From Figure (a), the degraded read time decreases with larger parameters for the reason that larger parameters lead to smaller-size blocks which perform better while adopting per-object coding. Overall, Optimal reduces the degraded read time of Flat and Random by 82.2% and 82.3%, 68.6% and 65.7%, 54.1% and 50.4%, 76.6% and 74.3%, 89.1% and 87.7%, respectively. From Figure (b), Optimal achieves 2.1× and 2.1×, 2.1× and 2.0×, 2.0× and 1.9×, 2.9× and 2.7×, 2.9× and 2.6×, under the five sets of parameters respectively. This validates the applicability of Optimal over different parameters.

From Figure (c), Optimal reduces the degraded read time of Flat and Random by 85.7% and 85.7%, 74.8% and 72.2%, 57.4% and 54.5%, 80.6% and 78.3%, 88.6% and 87.5%, respectively. From Figure (d), Optimal achieves 2.2× and 2.1×, 2.7× and 2.5×, 2.2× and 2.1×, 3.3× and 3.0×, 2.3× and 2.0×, under the five sets of parameters respectively.

**Experiment 3 (Repair performance under different cross-cluster bandwidths).** We now evaluate the repair performance under different cross-cluster bandwidths. We adopt (16, 12, 6) Azure-LRC and fix the object size as 4 MB. We vary the cross-cluster bandwidth from 0.5 Gbps to 10 Gbps (i.e., the ratio of inner-cluster bandwidth to cross-cluster bandwidth is 20:1 - 1:1). Figure followed shows the results.



（a）Degraded Read Time （b）Node Repair Rate

From Figure (a), the degraded read time decreases with a larger bandwidth. Besides, Optimal reduces the degraded read time of Flat and Random by 59.2% and 56.2%, 57.7% and 54.7%, 55.8% and 52.8%, 52.4% and 50.3%, 52.4% and 51.8%, when the bandwidth is 0.5 Gbps, 0.7 Gbps, 1 Gbps, 2 Gbps, 10 Gbps, respectively. This indicates that Optimal achieves larger gains under a scarcer cross-cluster bandwidth.

From Figure (b), the node repair rate increases with a larger bandwidth. Overall, Optimal achieves 2.3× and 2.2×, 2.2× and 2.1×, 2.1× and 2.0×,2.0× and 1.9×, and 1.9× and 2.0× node repair rate gains over Flat and Random, under the six bandwidth settings, respectively.

**Evaluation for Azure-LRC+1**

We evaluate our repair-optimal data placement (denoted by Optimal) via numerical analysis and testbed experiments. We compare Optimal with the flat data placement (denoted by Flat) which stores each block of a stripe in a distinct cluster , and the random data placement (denoted by Random) which stores the blocks at random subject to single-cluster fault tolerance.

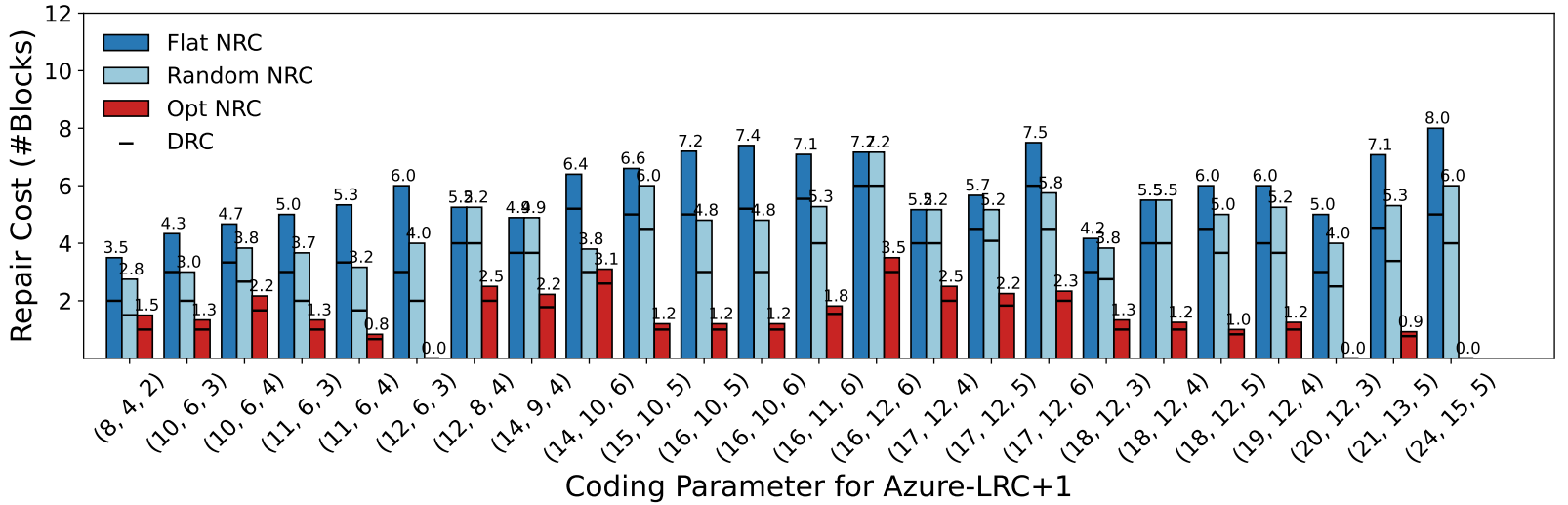
The overall results are summarized as follows. First, from numerical analysis, Optimal greatly reduces the cross-cluster repair traffic of Flat and Random. Second, from testbed experiments, Optimal reduces the degraded read time of Flat and Random by up to 87.6% and 86.6%, respectively, and achieves up to 8.3× and 7.8× node repair rate gains over Flat and Random, respectively.

**Numerical Analysis**

Comparisons of three data placements. We adopt two data placement schemes, i.e., flat data placement and random data placement, as the comparisons to measure the improvement that the optimal data placement achieved. The flat data placement is to put every block into a single cluster while the random data placement is to randomly choose random number of blocks and put them into a cluster subject to single-cluster fault tolerance.

For numerical analysis, we first consider the common parameters closed to Azure-LRC used in enterprise storage systems, for example (n = 15, k = 10, r = 5). For generalization, we consider the parameters, close to those commonly used, under 8 ≤ n ≤ 24, 4 ≤ k ≤ 15, 2 ≤ r ≤ 6. And we consider the redundancy ratio (i.e. n/k) from 1.33 to 2.

The Figure followed shows the repair costs of Flat, Random, and Optimal under 24 sets of coding parameter.



We summarize the following observations.

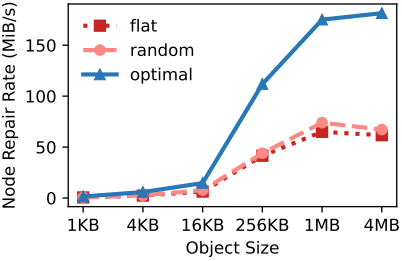
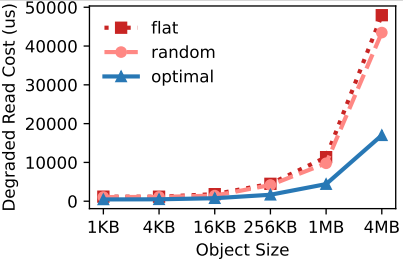
* Optimal always has the minimum NRC and DRC. Optimal reduces the NRC and DRC of Flat significantly, and also shows notable improvements over Random. For example, for (12, 6, 3) Azure-LRC+1, NRC = 6.0 and DRC = 3.0 in Flat, NRC = 4.0 and DRC = 2.0 in random, NRC = 0 and DRC = 0 in Optimal.
* For the case where d - 1 ≥ r + 1, Optimal places each local group of blocks in one cluster subject to single-cluster fault tolerance, the cost for repairing any block, implying NRC = 0 and DRC = 0. For example, for Azure-LRC+1 with parameters (20, 12, 3), (24, 15, 5), NRC = 0 and DRC = 0.
* As r increases, NRC and DRC increase in Flat, while they keep stable in Optimal. For example, for (17, 12, 4) Azure -LRC+1, NRC = 5.2 and DRC = 4.0 in Flat, and NRC = 2.5 and DRC = 2.0 in Optimal, while for (17, 12, 5) Azure-LRC+1, NRC = 5.7 and DRC = 4.5 in Flat, and NRC = 2.2 and DRC = 1.8 in Optimal, and for (17, 12, 6) Azure-LRC+1, NRC = 7.5 and DRC = 6.0 in Flat, and NRC = 2.3 and DRC = 2.0 in Optimal.
* NRC is always larger than DRC because the cost for repairing the parity blocks is amortized over the data blocks.

**Testbed Experiment**

Setup. We deploy our key-value store prototype on a local cluster which consists of five physical nodes, each of which runs Ubuntu 16.04.5 LTS with a quad-core 3.40 GHz Intel Core i5-3570, 16 GB RAM, and a Seagate ST1000DM003 7200 RPM 1 TB SATA hard disk. The nodes are connected by a 10 Gbps network. We assign one node as the network core and the remaining nodes as storage servers to emulate clusters. Each cluster consists of one proxy and multiple servers. We use the Wonder Shaper tool to control the outgoing bandwidth of the network core to make the cross-cluster bandwidth constrained, emulating cross-cluster transmission.

For testbed experiment, we adopt (n, k, r) = (15, 10, 5) Azure-LRC+1. We consider various object sizes from 1 KB to 4 MB and configure the ratio of the cross-cluster bandwidth to the inner-cluster bandwidth is 1:10, i.e. the bandwidth of network core is 1Gbps. We vary some of the settings in our experiments. We measure the degraded read time, which is the average time to repair all data blocks in a stripe. We then measure the node repair rate, which is the average rate to repair all nodes in a system. Note that node repair involves the repair of data and parity blocks. The results of each experiment are averaged over ten runs.

**Experiment 1 (Repair performance under different object sizes).** We first evaluate the repair performance of Optimal under different object sizes, varied from 1 KB to 4 MB. We consider (15, 10, 5) Azure-LRC+1, and adopt the default bandwidth configuration, i.e. 1 Gbps of cross-cluster bandwidth and 10 Gbps of inner-cluster bandwidth. The Figures followed shows the results.

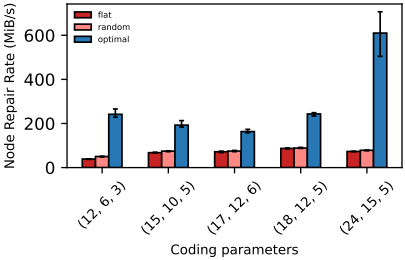
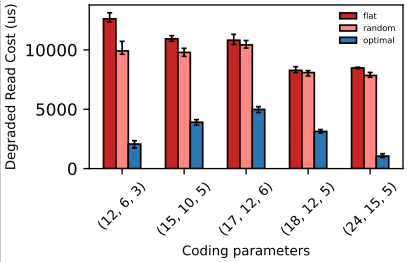


（a）Degraded Read Time （b）Node Repair Rate

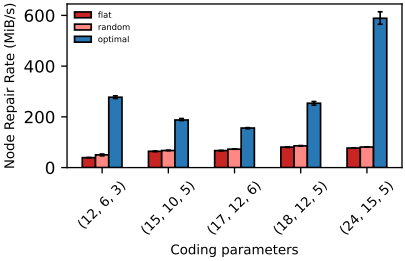
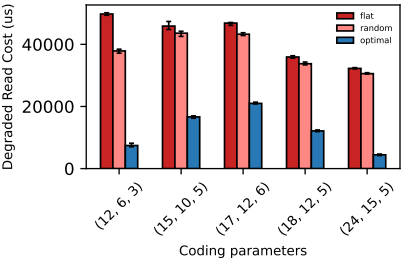
From Figure (a), the degraded read time increases with a larger object size, and the growth tends to flatten out as the object size becomes larger. And we can see that Optimal constantly outperforms Flat and Random. The experimental results comply with the theoretics, Optimal has a much smaller DRC than Flat and Random. Overall, Optimal reduces the degraded read time of Flat and Random by 60.7% and 56.7%, 57.5% and 55.3%, 57.4% and 46.3%, 62.9% and 60.2%, 61.2% and 54.9%, 64.6% and 60.9%, respectively, across all object sizes. We can see that Optimal achieves larger gains with a larger object size generally, as the cost for transferring blocks across clusters increases with a larger object size.

From Figure (b), the node repair rate also increases with a larger object size, while Optimal achieves larger node repair rates. Overall, Optimal achieves 2.5× and 2.2×, 2.4× and 2.3×, 2.3× and 1.9×, 2.7× and 2.5×, 2.7× and 2.4×, 2.9× and 2.7× node repair rate gains over Flat and Random, respectively, across all object sizes. Optimal also shows larger gains with a larger object size.

**Experiment 2 (Repair performance under different coding parameters).** We next evaluate repair performance under different coding parameters. We consider five sets of (n, k, r), i.e., (12, 6, 3), (15, 10, 5), (17, 12, 6), (18, 12, 5), (24, 15, 5). We consider two object sizes, 1MB and 4 MB. Figure followed shows the results (the error bars show the maximum and minimum results across over ten runs).



（a）Object size 1MB （b）Object size 1MB

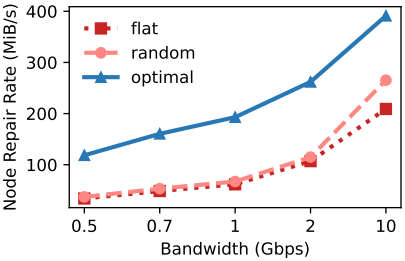
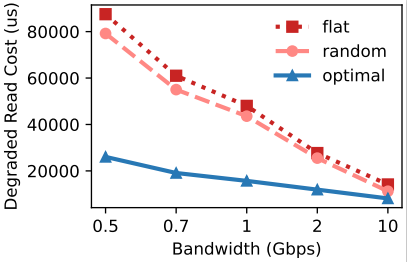


（c）Object size 4 MB （d）Object size 4 MB

From Figure (a), the degraded read time decreases with larger parameters for the reason that larger parameters lead to smaller-size blocks which perform better while adopting per-object coding. Overall, Optimal reduces the degraded read time of Flat and Random by 83.6% and 79.1%, 64.3% and 60.1%, 53.9% and 52.2%, 62.1% and 61.3%, 87.6% and 86.6%, respectively. From Figure (b), Optimal achieves 6.2× and 4.8×, 2.8× and 2.6×, 2.3× and 2.2×, 2.8× and 2.7×, 8.3× and 7.8×, under the five sets of parameters respectively. This validates the applicability of Optimal over different parameters.

From Figure (c), Optimal reduces the degraded read time of Flat and Random by 85.1% and 80.4%, 63.7% and 61.8%, 55.2% and 51.5%, 66.2% and 64.1%, 86.2% and 85.5%, respectively. From Figure (d), Optimal achieves 7.1× and 5.5×, 2.9× and 2.8×, 2.3× and 2.1×, 3.1× and 2.9×, 7.6× and 7.3×, under the five sets of parameters respectively.

**Experiment 3 (Repair performance under different cross-cluster bandwidths).** We now evaluate the repair performance under different cross-cluster bandwidths. We adopt (15, 10, 5) Azure-LRC+1 and fix the object size as 4 MB. We vary the cross-cluster bandwidth from 0.5 Gbps to 10 Gbps (i.e., the ratio of inner-cluster bandwidth to cross-cluster bandwidth is 20:1-1:1). Figure followed shows the results.



（a）Degraded Read Time （b）Node Repair Rate

From Figure (a), the degraded read time decreases with a larger bandwidth. Besides, Optimal reduces the degraded read time of Flat and Random by 70.2% and 67.1%, 68.7% and 65.3%, 67.3% and 64.0%, 56.9% and 53.2%, 42.5% and 27.6%, when the bandwidth is 0.5 Gbps, 0.75 Gbps, 1 Gbps, 2 Gbps, 10 Gbps, respectively. This indicates that Optimal achieves larger gains under ascarcer cross-cluster bandwidth.

From Figure (b), the node repair rate increases with a larger bandwidth. Overall, Optimal achieves 3.5× and 3.2×, 3.3× and 3.0×, 3.1× and 2.9×, 2.4× and 2.3×, and 1.9× and 1.5× node repair rate gains over Flat and Random, under the six bandwidth settings, respectively.