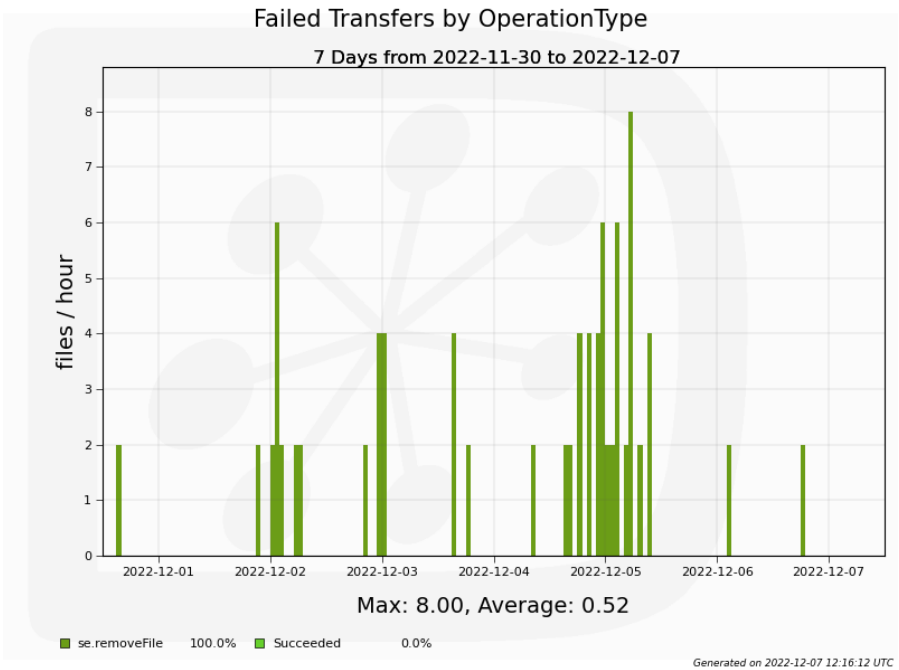
Deletion problems: current status

Problem

- Sometimes deletions from WNs on ECHO are failing
- Number of failures is very little for the last week, though the same is true for the number of jobs



Scenarios

- Long delete

Server side:

```
221206 18:27:39 ceph_posix_unlink : /lhcb:buffer/lhcb/MC/2018/SIM/00172177/0006/00172177_00067468_1.sim
221206 18:27:39 ceph_namelib : translated /lhcb:buffer/lhcb/MC/2018/SIM/00172177/0006/00172177_00067468_1.sim to
lhcb:buffer/lhcb/MC/2018/SIM/00172177/0006/00172177_00067468_1.sim
```

Client side:

```
2022-12-06 18:28:02 UTC dirac-jobexec/DIRAC.Resources.Storage.StorageElement/SE[RAL-BUFFER] VERBOSE: Failure in plugin to
perform removeFile Plugin: Echo lfn: /lhcb/MC/2018/SIM/00172177/0006/00172177_00067468_1.sim error Connection timed out ( 110
: GError('DavPosix::unlink  timeout of 20s', 110))
```

Scenarios

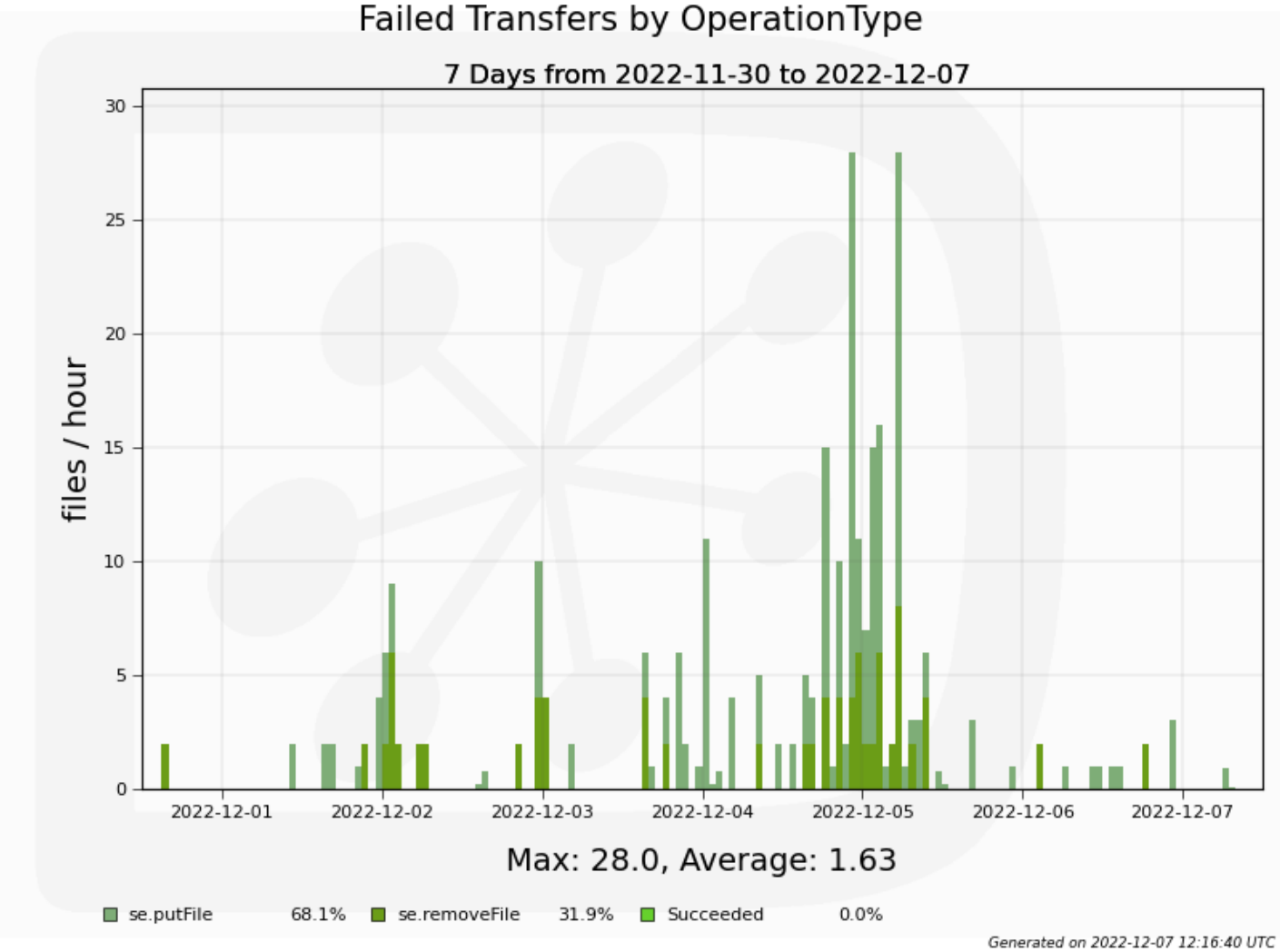
- Client disappears

Client side:
Same timeout or:

```
2022-12-05 05:38:15 UTC dirac-jobexec/DIRAC.Resources.Storage.StorageElement/SE[RAL-BUFFER] VERBOSE: Failure in plugin to
perform putFile Plugin: Echo lfn: /lhcb/MC/2018/KSTARTAUTAU.STRIP.DST/00172524/0000/00172524_00000173_7.KstarTauTau.Strip.dst
error Too many open files ( 24 : Failed to copy file
/pool/condor/dir_3647501/ZeeLDmNO3L2nCIXDjqiBL5XqABFKDmABFKDm14iPDmABFKDmtc2cNm/DIRAC_86rYP4pilot/688183994/00172524_00000173
_7.KstarTauTau.Strip.dst to destination url
https://webdav.echo.stfc.ac.uk:1094/lhcb:buffer/lhcb/MC/2018/KSTARTAUTAU.STRIP.DST/00172524/0000/00172524_00000173_7.KstarTau
Tau.Strip.dst: [24] TRANSFER ERROR: Copy failed (streamed). Last attempt: curl error (35): SSL connect error (destination))
[..]
2022-12-05 05:38:24 UTC dirac-jobexec/DIRAC.Resources.Storage.StorageElement/SE[RAL-BUFFER] VERBOSE: Failure in plugin to
perform removeFile Plugin: Echo lfn: /lhcb/MC/2018/SIM/00172523/0000/00172523_00001782_1.sim error Permission denied ( 13 :
GError('DavPosix::unlink  curl error (35): SSL connect error', 13))
```

Server side: just Macaroon request

Some of the deletion and put failures may have the same reason.



Vector read: current status

Problem

- Vector read requests to ECHO are slow. Sometimes so slow that the timeout is exceeded.

Naive approach

- The maximum number of chunks in a single vector read request (max_iov) can be limited
- The smaller number of chunks, the faster request is executed
- Client can query max_iov value from the server and adapt its behavior accordingly
- LHCb client (Gaudi) respects max_iov value

So we can limit max_iov (the patch is ready), to prevent the timeouts. This is only a mitigation.

Current implementation

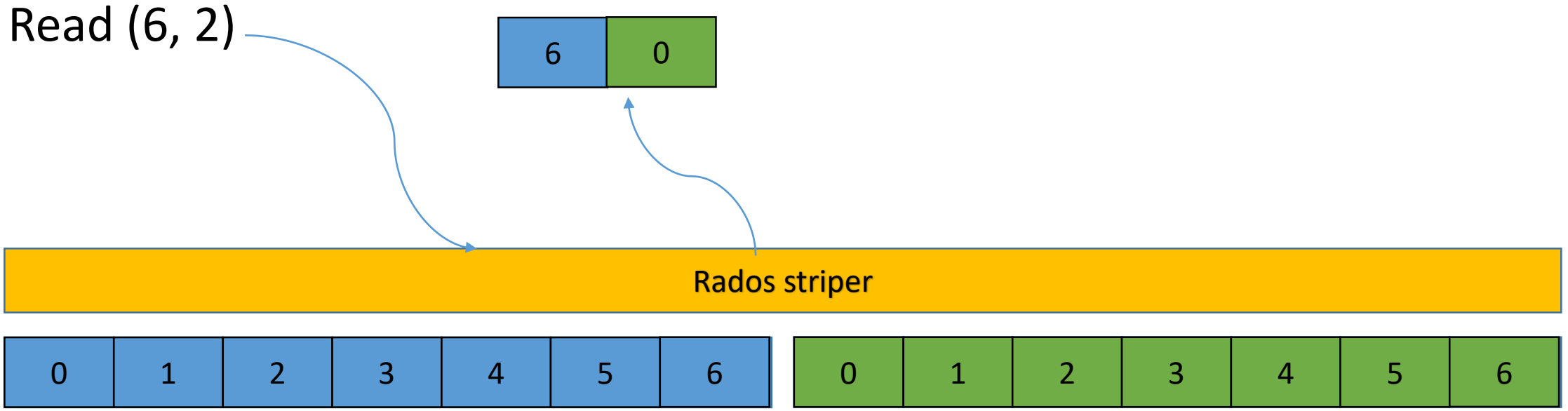
```
ssize_t XrdCephOssFile::ReadV(XrdOucIOVec *readV, int n)
{
   ssize_t nbytes = 0, curCount = 0;
   for (int i=0; i<n; i++)
       {curCount = Read((void *)readV[i].data,
                        (off_t)readV[i].offset,
                       (size_t)readV[i].size);
        if (curCount != readV[i].size)
           {if (curCount < 0) return curCount;
            return -ESPIPE;
           }
        nbytes += curCount;
       }
   return nbytes;
}
```

Current implementation

- Reads are sequential (there is an async implementation, does it work?)
- Every time read is called, new connection context is created (though there may be a connection caching)
- Rados striper is used

How files are stored at RAL's ECHO SE?

Every file is split into objects, each object is 64MiB in size (except the last one?). So, if a file dir/file1 is 128MiB, there are, in fact, two objects in ceph: dir/file1.0000000000000000 and dir/file1.0000000000000001. Library radosstriper allow one to handle this transparently. And probably does much more than this.



Merging reads into a single function

- Instead of calling multiple high-level reads we can create context once, and then execute all reads with striper
- Pre-preliminary tests show that this does not affect performance

Caching

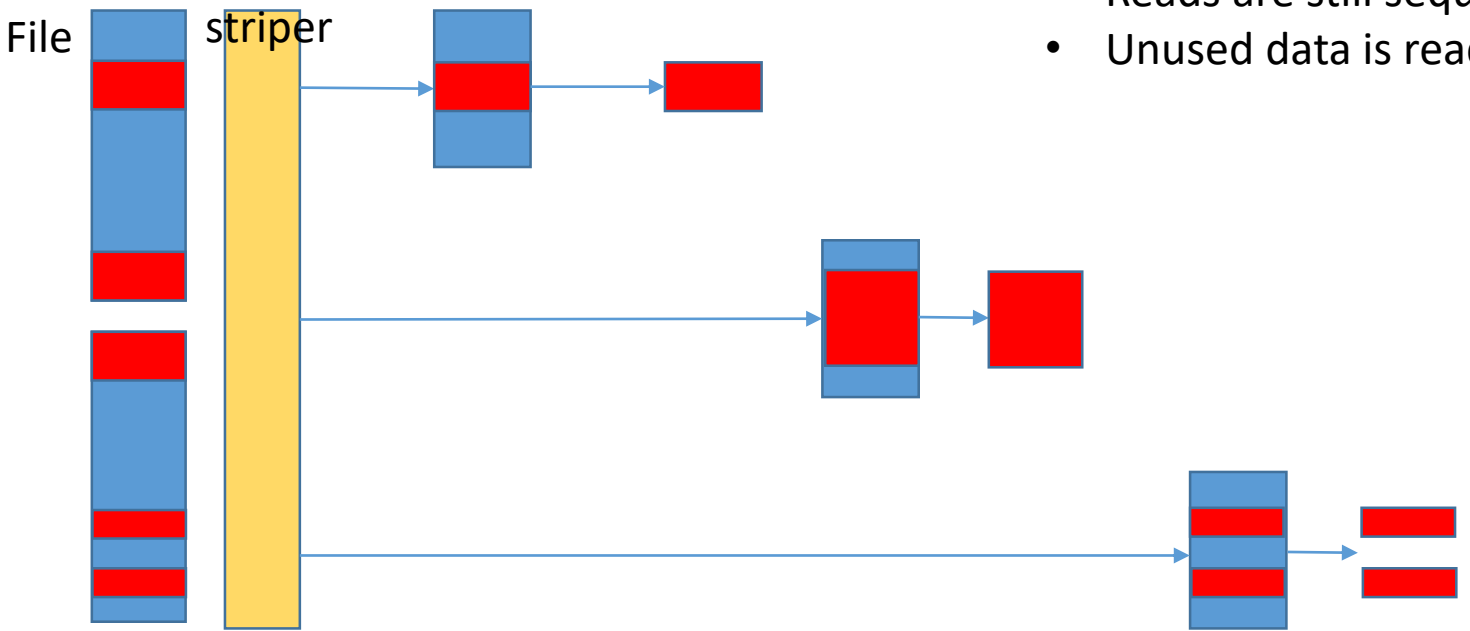
We can read huge blocks of data from storage using radosstriper and extract readv chunks from these blocks

Pros

- It may work

Cons

- Extra memory required
- Change of “coordinates”, potential errors
- Reads are still sequential
- Unused data is read

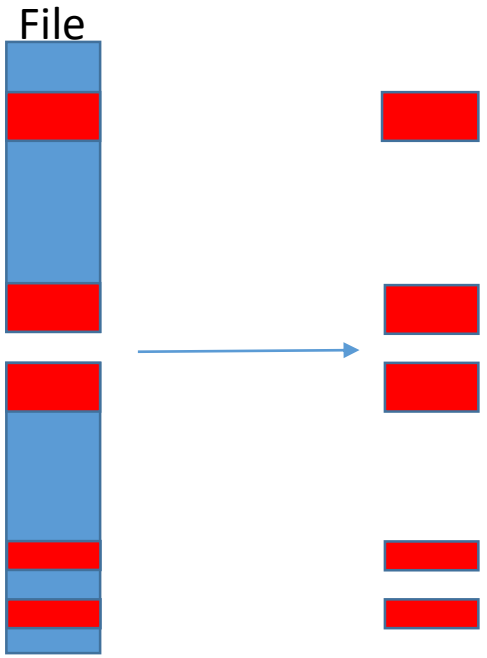


Removing striper

It is possible to read blocks directly from objects

Pros
- Async reads
- No extra memory

Cons
- Change of coordinates



Functional tests

Since we are refurbishing readv implementation, it’s a good idea to have functional tests. The simplest (probably) one is to compare readv results with sequential reads (see link above).

Preliminary results

Here are preliminary results of streaming 1 file (“benchmark”) using lhcb software. The results are obtained from external(!) gateway, which was not in production, so free of load.

Test type	Limit max_iov	Caching	No striper	No changes (different gw with similar config and huge timeout)
Time	325m47.778s	~10m	11m0.818s	323m1.111s

To do

- Try the changes on WN with local gateway