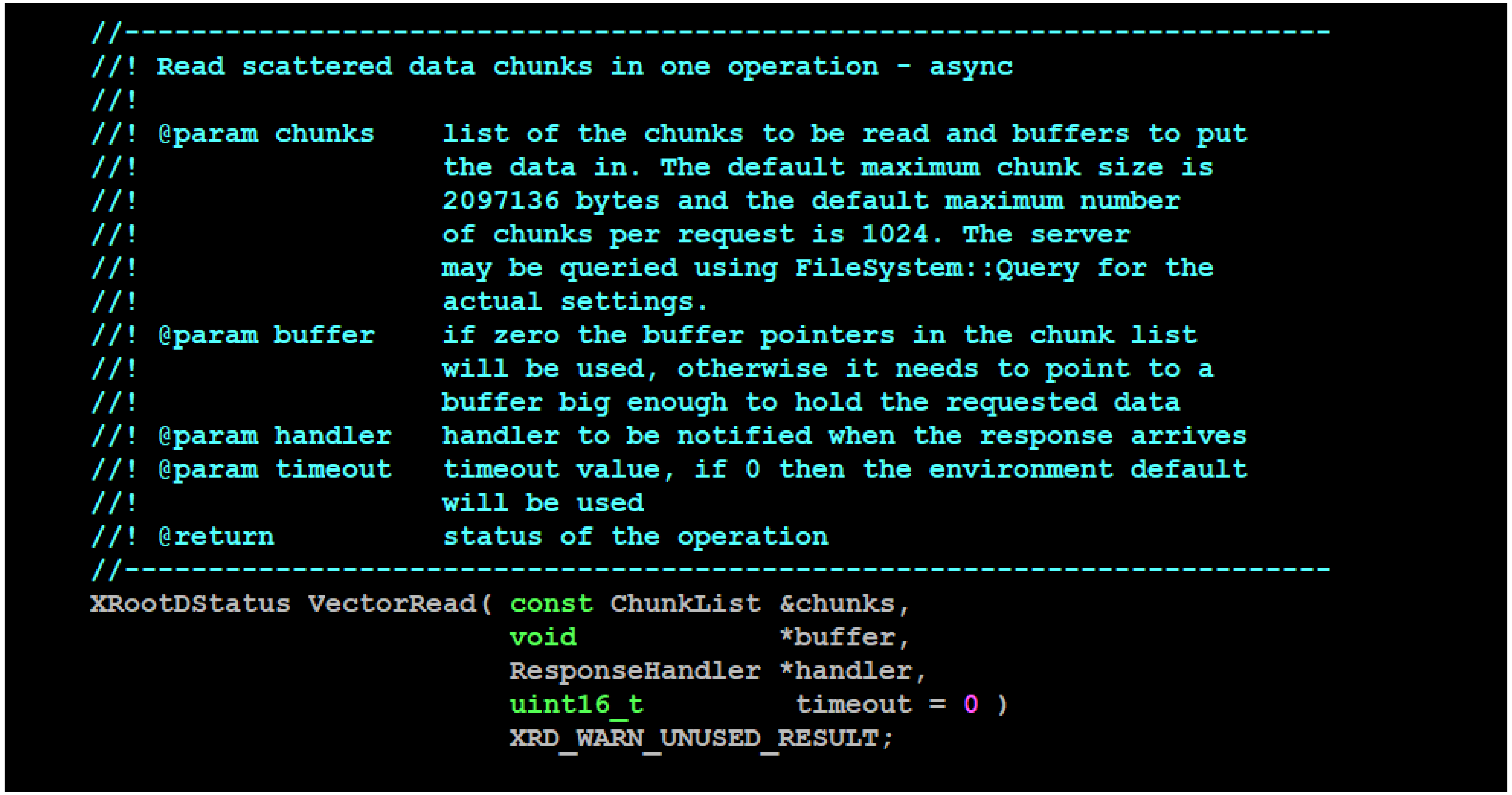
Current implementation and testing of Vector Reads in the XRootD Ceph plugin

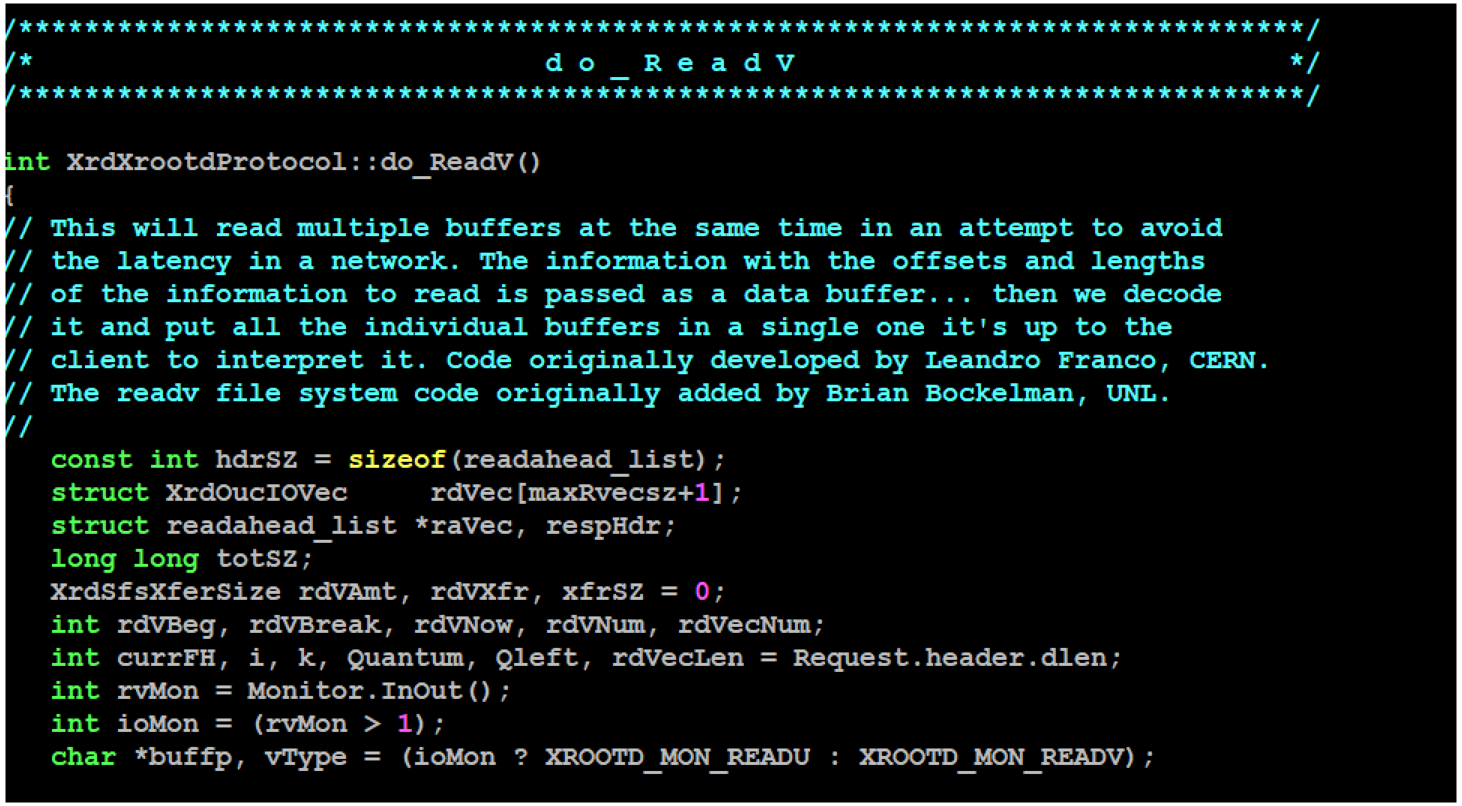
1. A vector read is triggered by the client application calling *XrdCl::File::VectorRead* (XrdCl/XrdClFile.hh)



```
//------------------------------------------------------------------------
//! Read scattered data chunks in one operation - async
//!
//! @param chunks    list of the chunks to be read and buffers to put
//!                  the data in. The default maximum chunk size is
//!                  2097136 bytes and the default maximum number
//!                  of chunks per request is 1024. The server
//!                  may be queried using FileSystem::Query for the
//!                  actual settings.
//! @param buffer    if zero the buffer pointers in the chunk list
//!                  will be used, otherwise it needs to point to a
//!                  buffer big enough to hold the requested data
//! @param handler   handler to be notified when the response arrives
//! @param timeout   timeout value, if 0 then the environment default
//!                  will be used
//! @return          status of the operation
//------------------------------------------------------------------------
XRootDStatus VectorRead( const ChunkList &chunks,
                         void             *buffer,
                         ResponseHandler *handler,
                         uint16_t          timeout = 0 )
                         XRD_WARN_UNUSED_RESULT;
```

Translates into a kxR_readv request

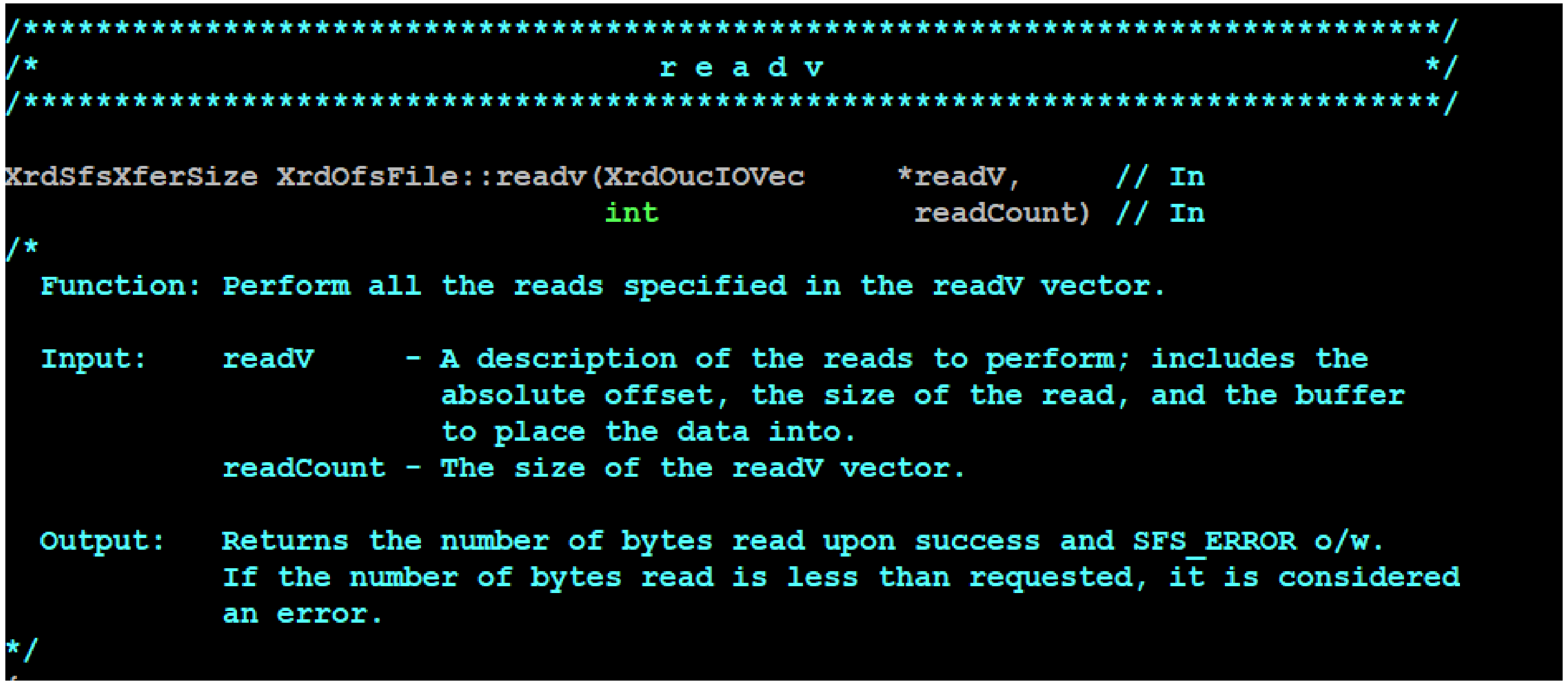
2. The vector read request is handled by the XRootD server in XrdXrootd/XrdXrootdXeq.cc



```
/******************************************************************************/
/*                              d o _ R e a d V                               */
/******************************************************************************/

int XrdXrootdProtocol::do_ReadV()
{
// This will read multiple buffers at the same time in an attempt to avoid
// the latency in a network. The information with the offsets and lengths
// of the information to read is passed as a data buffer... then we decode
// it and put all the individual buffers in a single one it's up to the
// client to interpret it. Code originally developed by Leandro Franco, CERN.
// The readv file system code originally added by Brian Bockelman, UNL.
//
   const int hdrSZ = sizeof(readahead_list);
   struct XrdOucIOVec     rdVec[maxRvecsz+1];
   struct readahead_list *raVec, respHdr;
   long long totSZ;
   XrdSfsXferSize rdVAmt, rdVXfr, xfrSZ = 0;
   int rdVBeg, rdVBreak, rdVNow, rdVNum, rdVecNum;
   int currFH, i, k, Quantum, Qleft, rdVecLen = Request.header.dlen;
   int rvMon = Monitor.InOut();
   int ioMon = (rvMon > 1);
   char *buffp, vType = (ioMon ? XROOTD_MON_READU : XROOTD_MON_READV);
```

https://github.com/xrootd/xrootd/blob/master/src/XrdXrootd/XrdXrootdXeq.cc#L2439

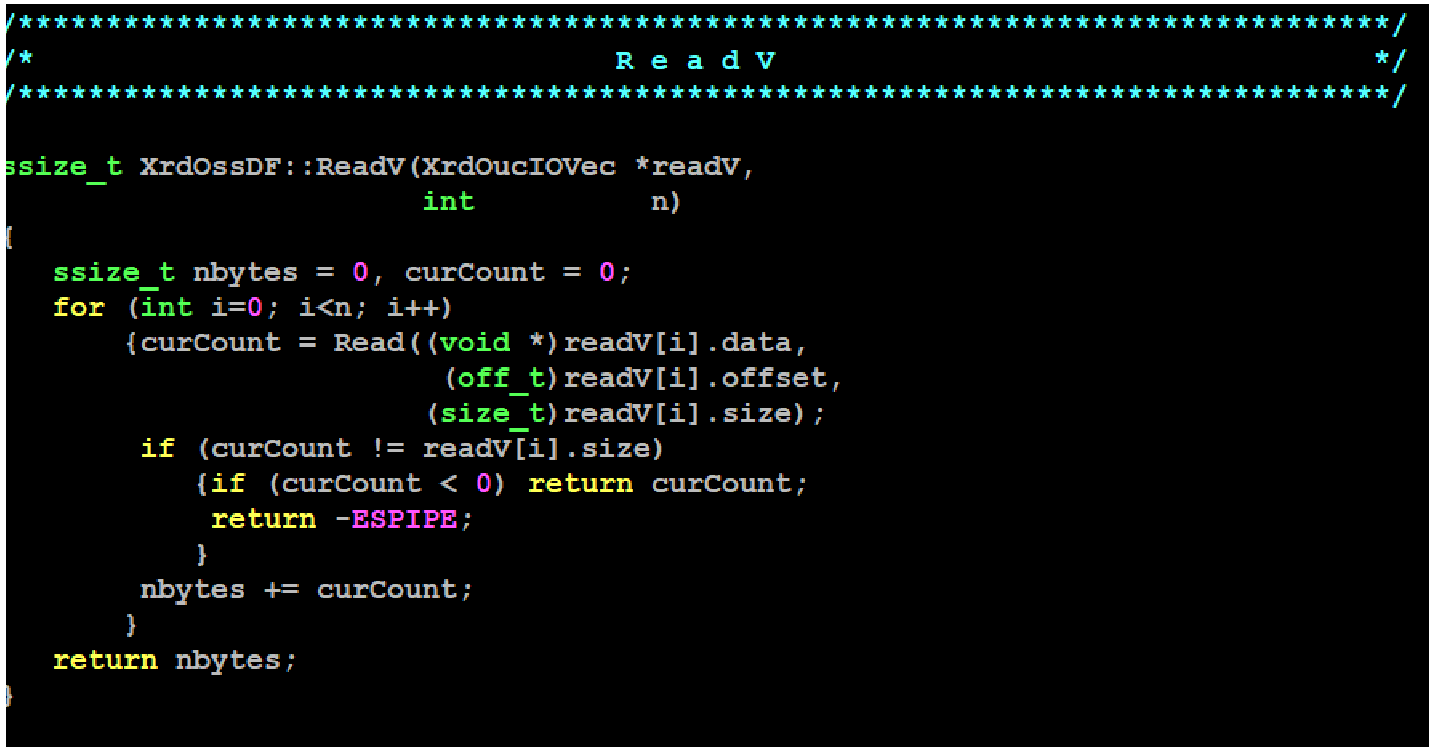
3. The server will use the OFS plug-in method *readv*:



```
/******************************************************************************/
/*                                 r e a d v                                  */
/******************************************************************************/

XrdSfsXferSize XrdOfsFile::readv(XrdOucIOVec     *readV,     // In
                                 int              readCount) // In
/*
  Function: Perform all the reads specified in the readV vector.

  Input:    readV     - A description of the reads to perform; includes the
                        absolute offset, the size of the read, and the buffer
                        to place the data into.
            readCount - The size of the readV vector.

  Output:   Returns the number of bytes read upon success and SFS_ERROR o/w.
            If the number of bytes read is less than requested, it is considered
            an error.
*/
```

https://github.com/xrootd/xrootd/blob/master/src/XrdOfs/XrdOfs.cc#L1099

4. Finally, the OFS plug-in calls the *ReadV* method at the OSS layer:



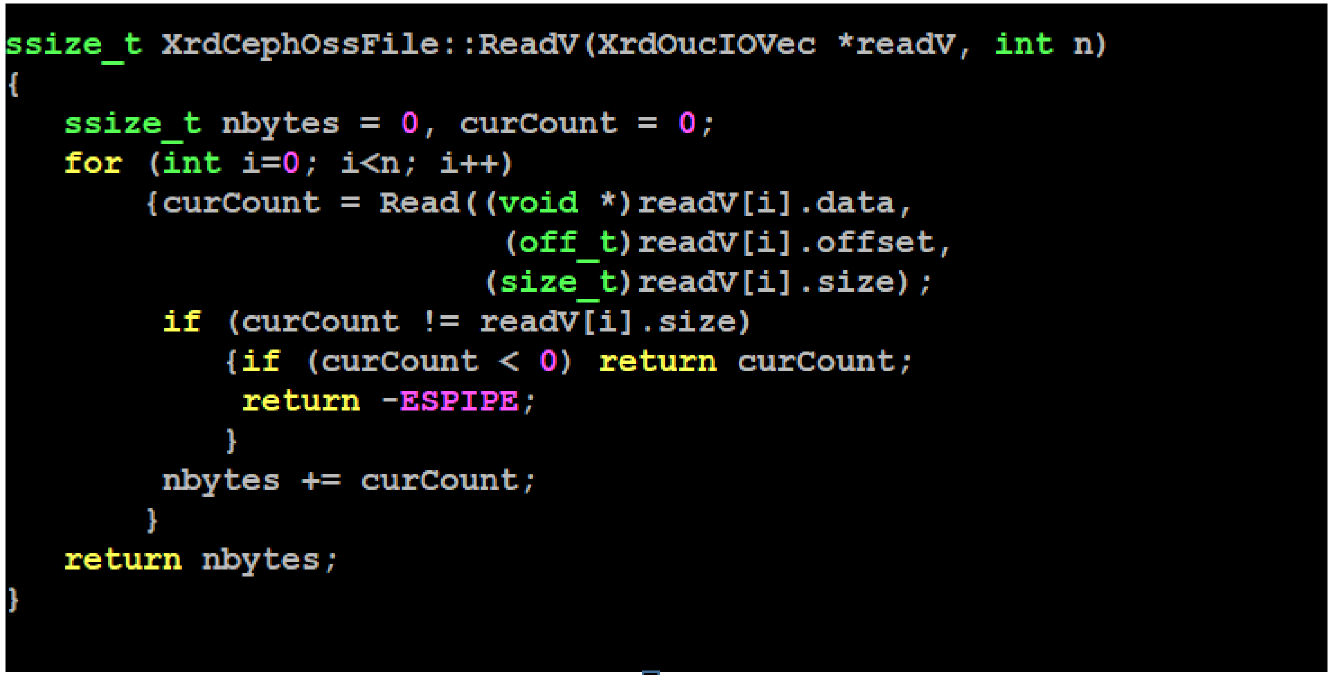
```
/******************************************************************************/
/*                                R e a d V                                   */
/******************************************************************************/

ssize_t XrdOssDF::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;
}
```

https://github.com/xrootd/xrootd/blob/master/src/XrdOss/XrdOss.cc#L257-L266

loops over XrdOucIOVec and calls a synchronous read for each buffer in the vector

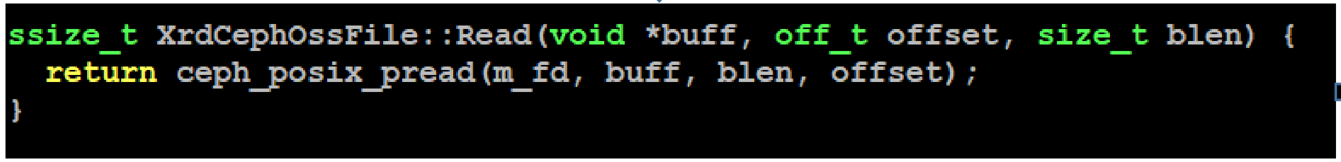
Because the OSS layer is implemented by the Ceph plug-in which does not define a ReadV method, Ian overloaded ReadV in XrdCeph/XrdCephOssFile.cc

Current implementation of *ReadV* in the XRootD Ceph plugin



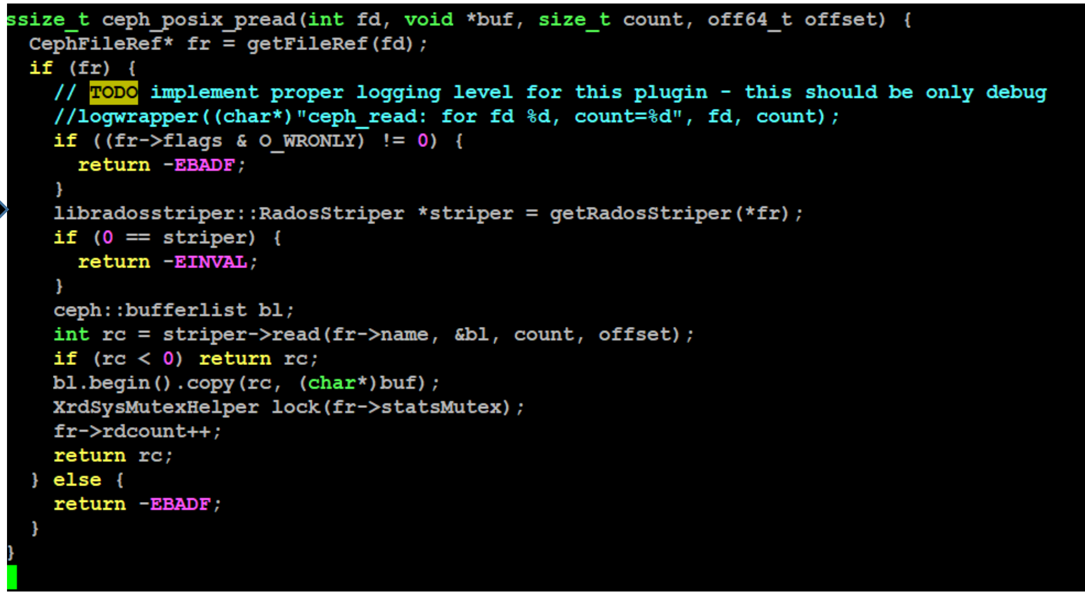
```
ssize_t XrdCephOssFile::ReadV(XrdOucIOVec *readV, int n)
{
   ssize_t nbytes = 0, curCount = 0;
   for (int i=0; i<n; i++)
       {curCount = Read((void *)readV[i].data,
                          (off_t)readV[i].offset,
                         (size_t)readV[i].size);
        if (curCount != readV[i].size)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

https://github.com/stfc/xrootd-ceph/commit/a68162f981227bdead0d3d5481b7730aab0e5cf7



```
ssize_t XrdCephOssFile::Read(void *buff, off_t offset, size_t blen) {
  return ceph_posix_pread(m_fd, buff, blen, offset);
}
```

https://github.com/stfc/xrootd-ceph/blob/vector_read/src/XrdCeph/XrdCephPosix.cc#L909



```
ssize_t ceph_posix_pread(int fd, void *buf, size_t count, off64_t offset) {
  CephFileRef* fr = getFileRef(fd);
  if (fr) {
    // TODO implement proper logging level for this plugin - this should be only debug
    //logwrapper((char*)"ceph_read: for fd %d, count=%d", fd, count);
    if ((fr->flags & O_WRONLY) != 0) {
      return -EBADF;
    }
    libradosstriper::RadosStriper *striper = getRadosStriper(*fr);
    if (0 == striper) {
      return -EINVAL;
    }
    ceph::bufferlist bl;
    int rc = striper->read(fr->name, &bl, count, offset);
    if (rc < 0) return rc;
    bl.begin().copy(rc, (char*)buf);
    XrdSysMutexHelper lock(fr->statsMutex);
    fr->rdcount++;
    return rc;
  } else {
    return -EBADF;
  }
}
```

The above implementation of ReadV is deployed on one of the test Echo gateways (ceph-test-gw691)

The test ran ROOT application making a call to TTree::GetEntry() and from then on it is all hidden

LHCb (Raja) run a test against this gateway

```
Error in <TNetXNGFile::ReadBuffers>: [ERROR] Operation expired
Error in <TBranchElement::GetBasket>: File: root://ceph-test-
gw691.gridpp.rl.ac.uk/lhcb:prod/lhcb/LHCb/Collision15/TURBO.MDST/00051289/0000/00051289_00003811_1.turbo.
mdst at byte:5285097, branch:_Event., entry:100, badread=1, nerrors=1, basketnumber=2
Error in <TBasket::Streamer>: The value of fNbytes is incorrect (-852418181) ; trying to recover by setting it to zero
Error in <TBranchElement::GetBasket>: File: root://ceph-test-
gw691.gridpp.rl.ac.uk/lhcb:prod/lhcb/LHCb/Collision15/TURBO.MDST/00051289/0000/00051289_00003811_1.turbo.
mdst at byte:-4297358380894729972, branch:_Event., entry:100, badread=1, nerrors=2, basketnumber=2
Error in <TBasket::Streamer>: The value of fNbytes is incorrect (-852418181) ; trying to recover by setting it to zero
Error in <TBranchElement::GetBasket>: File: root://ceph-test-
gw691.gridpp.rl.ac.uk/lhcb:prod/lhcb/LHCb/Collision15/TURBO.MDST/00051289/0000/00051289_00003811_1.turbo.
mdst at byte:-4297358380894729972, branch:_Event., entry:100, badread=1, nerrors=3, basketnumber=2
```

The ROOT method TNetXNGFile::ReadBuffers (line 469) is making the call to XrdCl::File::VectorRead (line 545)

https://root.cern.ch/doc/master/TNetXNGFile_8cxx_source.html