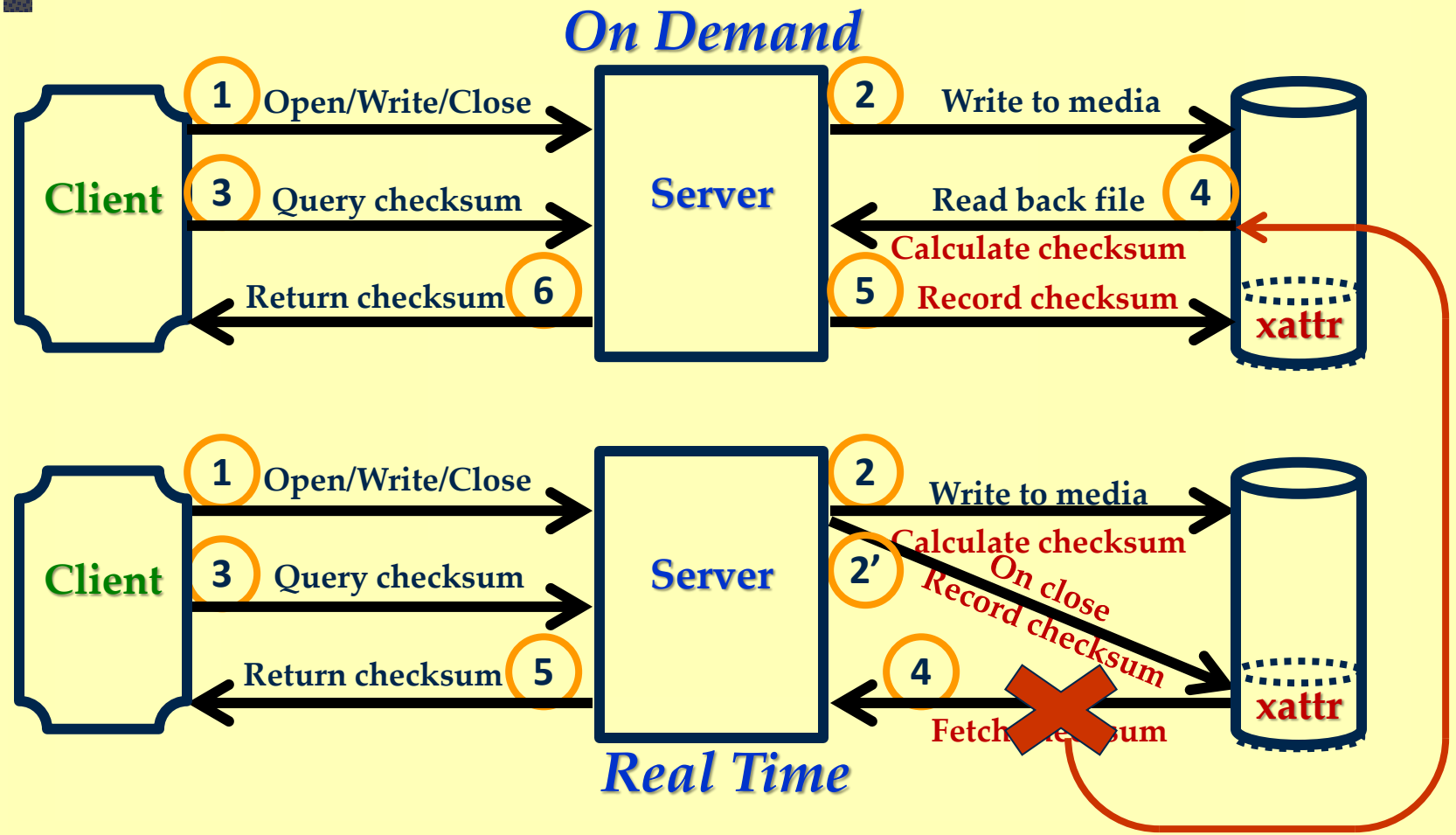
XRootD Real-time Streaming Checksums

Atlas S&C Week #84

June 29 – July 3, 2026

Andrew Hanushevsky, SLAC

On Demand vs. Real Time

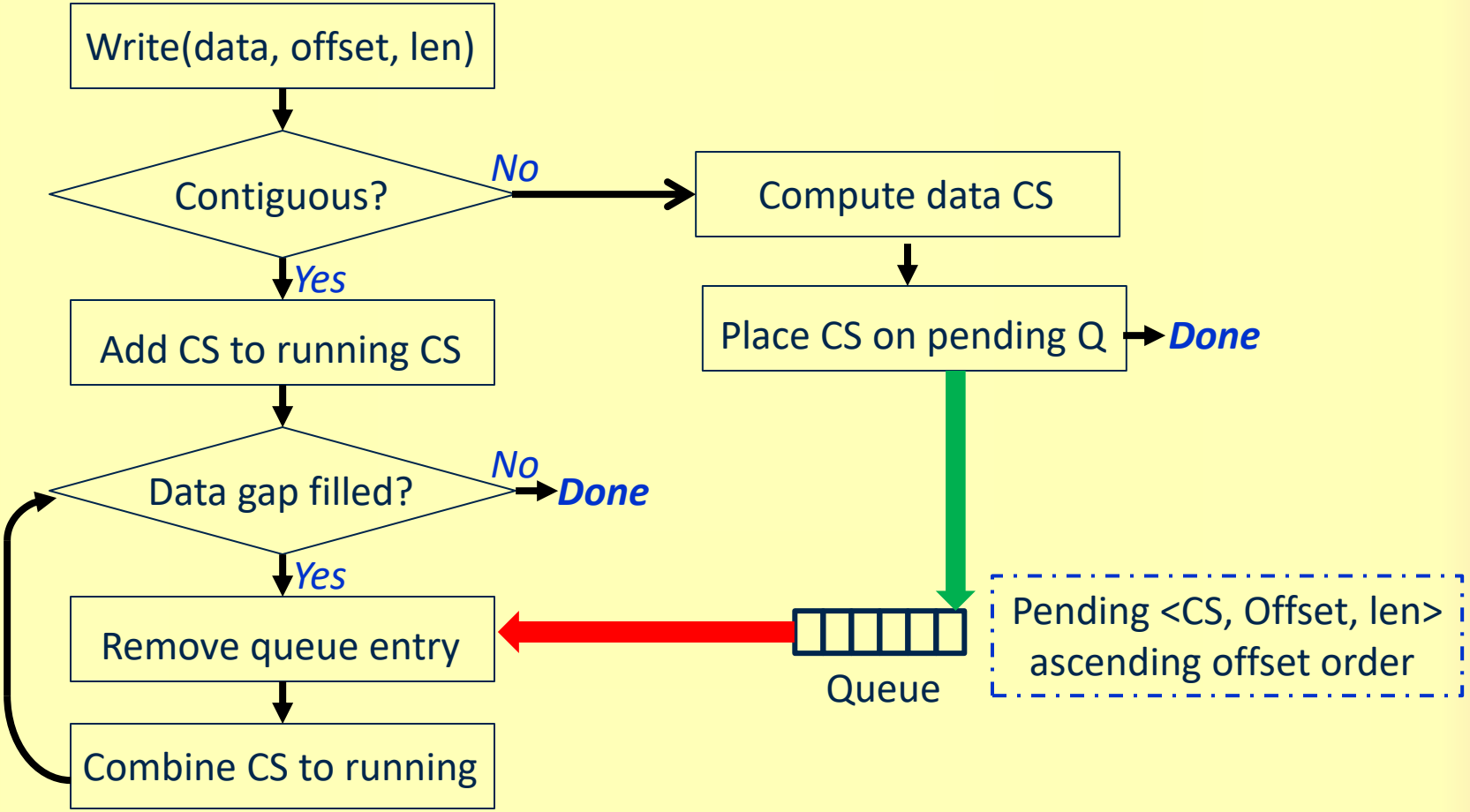


Out of order writes

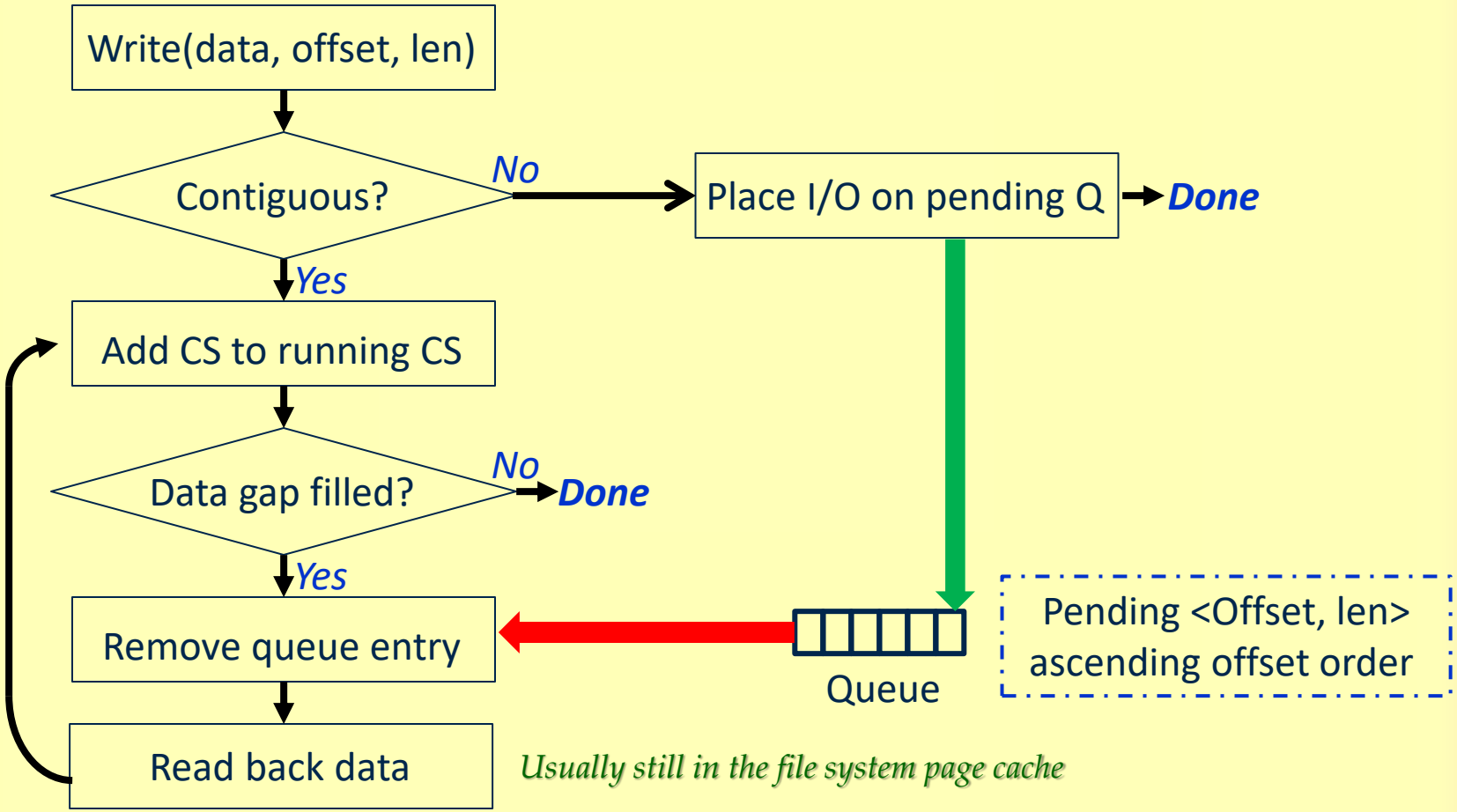
- Multi-streaming undermines write order
  - Stream I/O is asynchronous
    - This is true for the client as well as the server
  - Not an issue for on demand checksums
- Real time checksums handle write order
  - This is occurs for any kind of checksum
    - However, efficiency is checksum dependent
    - High efficiency for combinable checksums
      - I.E. cs_1[data1] *combine* cs_2[data2] = cs_3[data1||data2]
      - Polynomial CRC's and adler32

SLAC NATIONAL ACCELERATOR LABORATORY

Combinable ordering (simplified)



Noncombinable ordering (simplified)



On Demand vs. Real Time

Feature	On Demand	Real Time
CPU efficiency	Good	Good (Great for EC Ceph)
I/O Efficiency	Low (Reread is needed)	High (Reread not needed)
Complexity	Low	Intermediate (Must track writes)
Predictability (Multi-stream copy)	High	Low (Order dependent)
Integrity (Trust in checksum)	High	Low (Ignores disk errors)

SLAC NATIONAL ACCELERATOR LABORATORY

So, why is real time better?

- Unsurpassed in efficiency
  - 50% less resource use in many dimensions
    - Media contention
    - I/O buss utilization
    - CPU use for erasure encoded file systems
- However …
  - Disk errors are caught indirectly downstream
    - This can have real impacts on applications
      - Random failures due to uncaught media errors

Enabling real-time checksums

- Use the **ofs.cksrt** directive
  - **ofs.cksrt** [**auto** {**default** | **off** | *cipher*}] [**chkcgi**]
    - **auto** automatically performs real-time checksum if
      - Creating or replacing a file with open in write mode
    - **default** uses checksum specified via **xrootd.chksum**
      - *cipher* uses the specified native checksum
        - Native checksums: **adler32**, **crc32**, **crc32c**, and **md5**
    - **off** disables feature which is the default
    - **chkcgi** allows applications to control r/t checksum
      - Open **URL** must contain the **CGI** "**cks.type**=*cipher*"

SLAC NATIONAL ACCELERATOR LABORATORY

Things that can go wrong

- A real-time checksum fails and abandoned when:
  - Any part of the file is overwritten
  - Any part of the file remains unwritten on close
  - File is truncated with a non-zero length
  - File is subject to a vector write
  - An I/O or other error occurs
  - The file is not closed before client disconnects

Do bad things really happen?

- Not too often but when it happens …
  - Checksum recovery is inapplicable
    - Resume where you left off w/o real-time checksum
      - Checksum recovery tremendously increases complexity
  - Warning message printed to log with filename
    - Includes reason why real-time checksum failed
  - Real-time checksum not recorded
  - File transitions to an on-demand checksum

SLAC NATIONAL ACCELERATOR LABORATORY

Availability

- XRootD 6.1.0
  - Protocols
    - xroot
    - http/davs
    - TPC all protocols, pull and push
  - Multi-stream supported in all protocols
    - Out-of-order writes are transparently handled
- Additional improvements expected in 6.2
  - Faster computation for crc32*x* checksums