中国科学院高能物理研究所

A scheme to implement local server computation on EOS system based on Xrootd plug-in

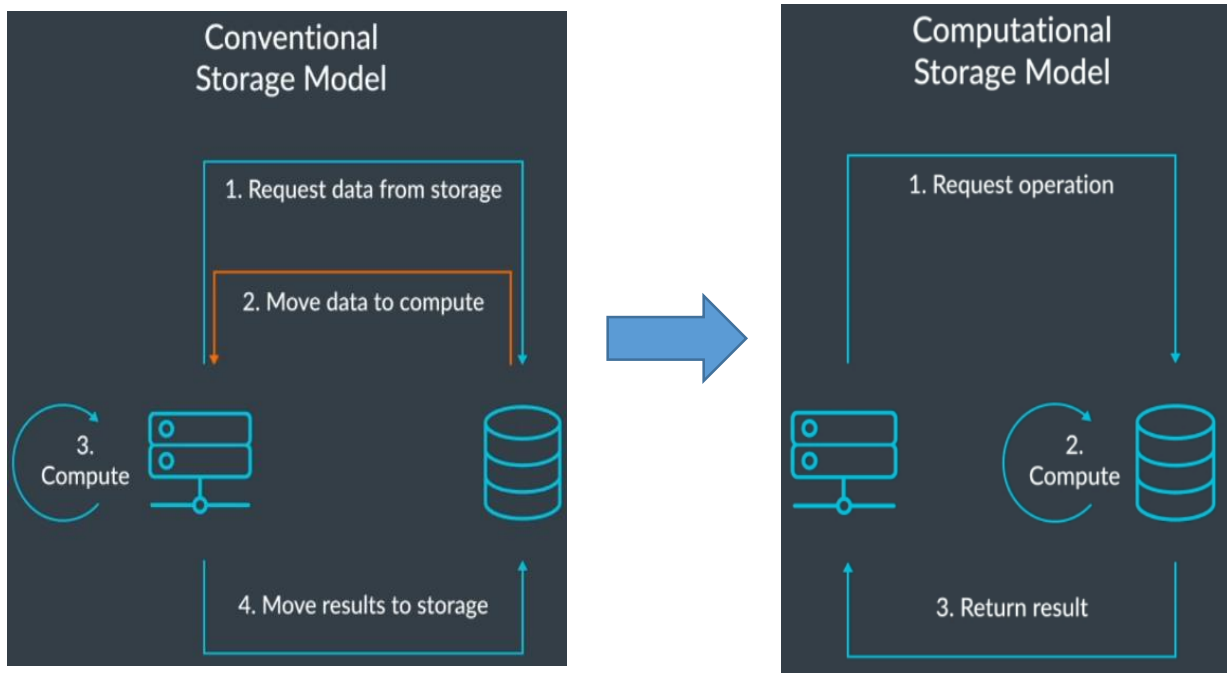
Speaker: Minxing Zhang

Motivation

- **Large scale high energy physics experiments will generate hundreds of PB data**
  - need better I/O bandwith
- **The "storage wall" problem caused by the separation of storage and computation in classical von Neumann architecture**
  - need less amounts of data moved between storage and computing platforms
- **Using a computational storage architecture can effectively reduce data movement**
  - Reduce network transmission, reduce cross-node communication

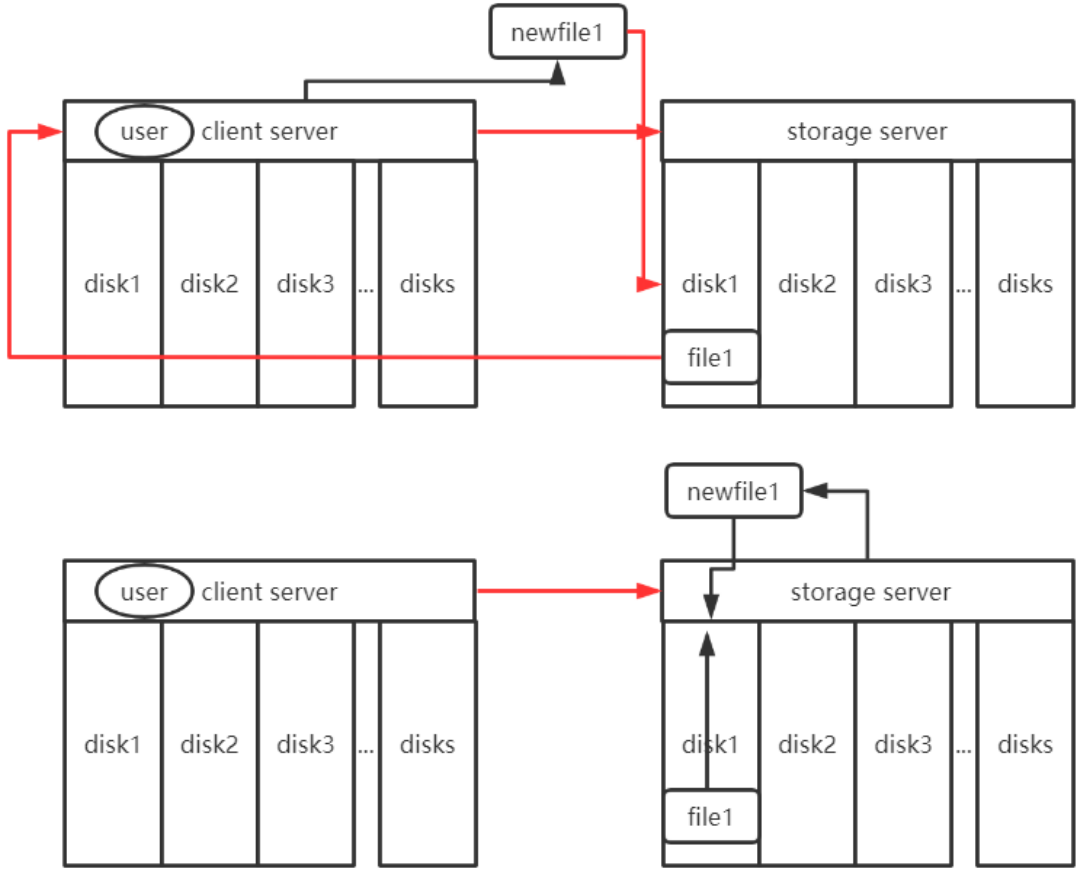
Related Studies

- Computational Storage
  - CSS
    - Local Computing Services
    - At the software level
  - CSP
    - The function of transparency to the upper layer
    - Located on the hard drive
  - CSD
    - FPGA
    - Matrix operation, encryption operation, etc
  - CSA
    - The computational storage nodes that make up the array



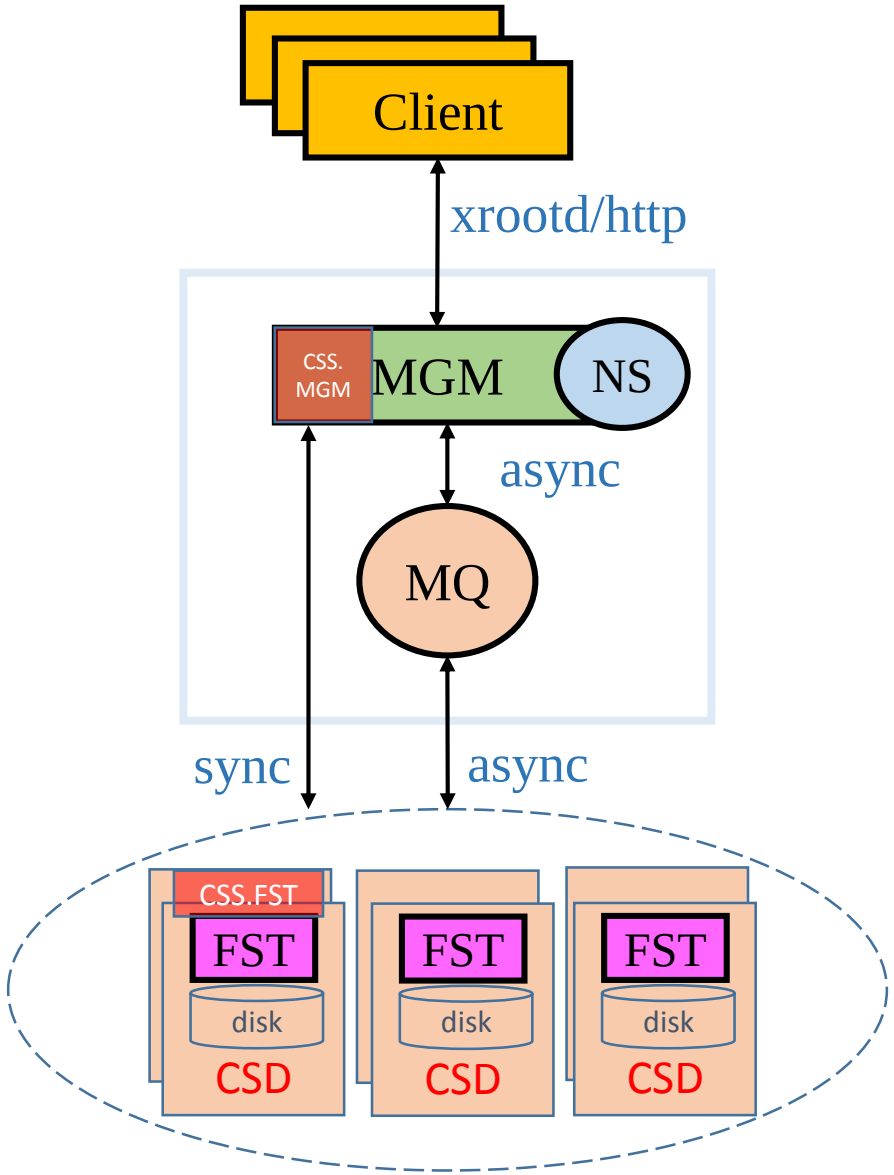
System Design

- Traditional distributed systems use files for computing:
  - Three network communications
  - Two large file transmission
- These calculations are performed using computational storage
  - One network communication
  - No large file transmission



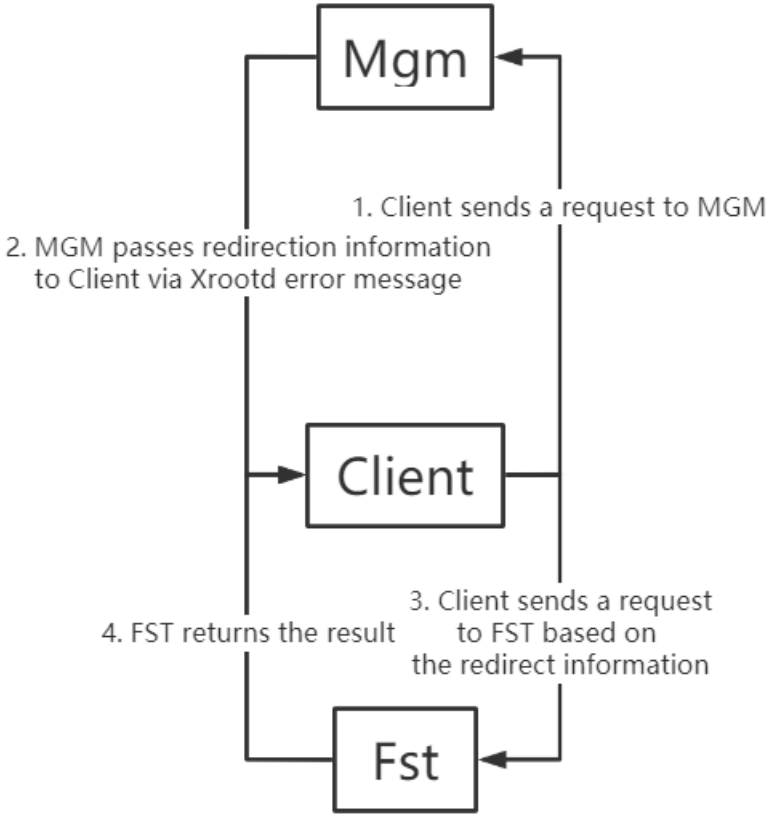
System Design

- Based on EOS and Xrootd implementations
- In general, it still looks like an EOS storage system, reading and writing data normally
- When the function needs to be called, the user appends a special parameter to the Open path
- The corresponding new file is then generated locally at the FST



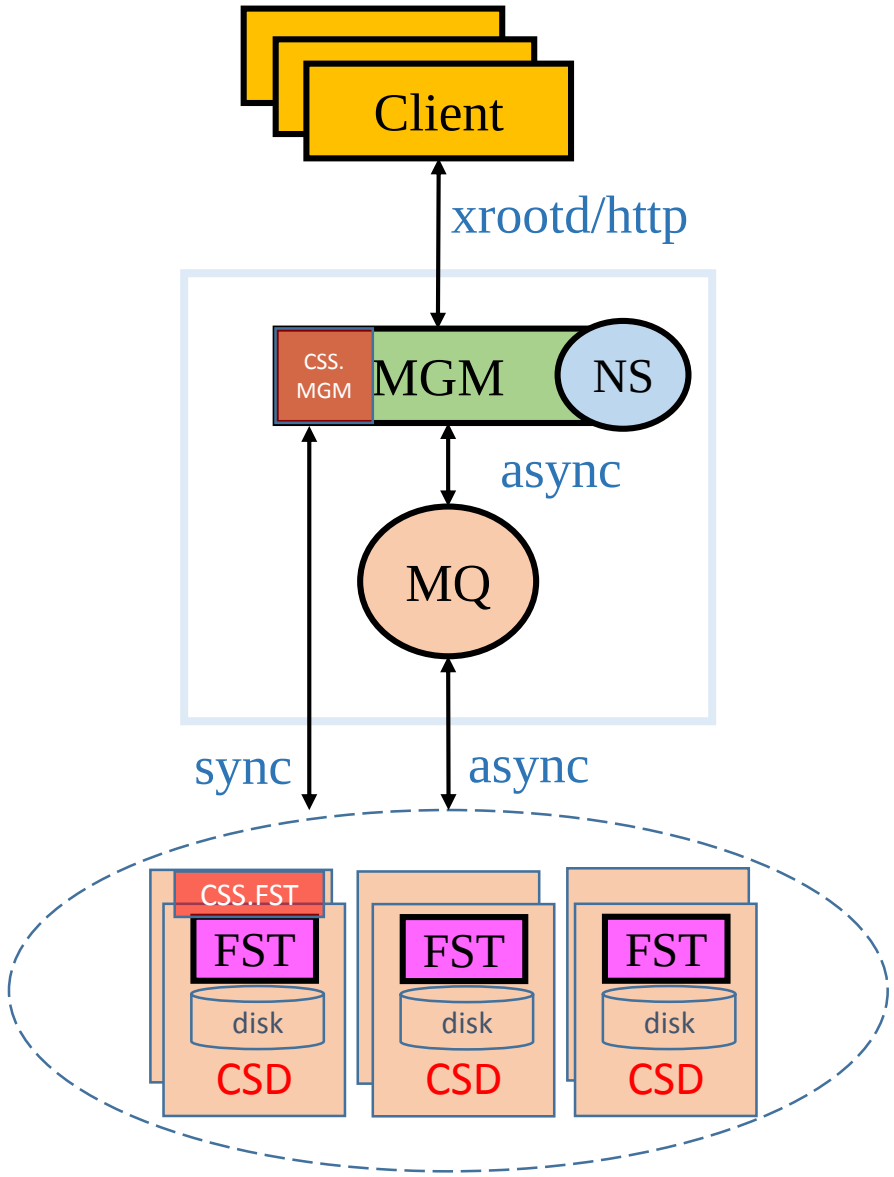
System Design

- The process of opening a file in EOS is generally as follows:
  - The client receives an access request and sends a list of queries to the MGM.
  - The MGM then checks the metadata, finds the FST where the file resides, and returns the client redirect result.
  - The client receives the redirect request and forwards it to the FST where the file is located.

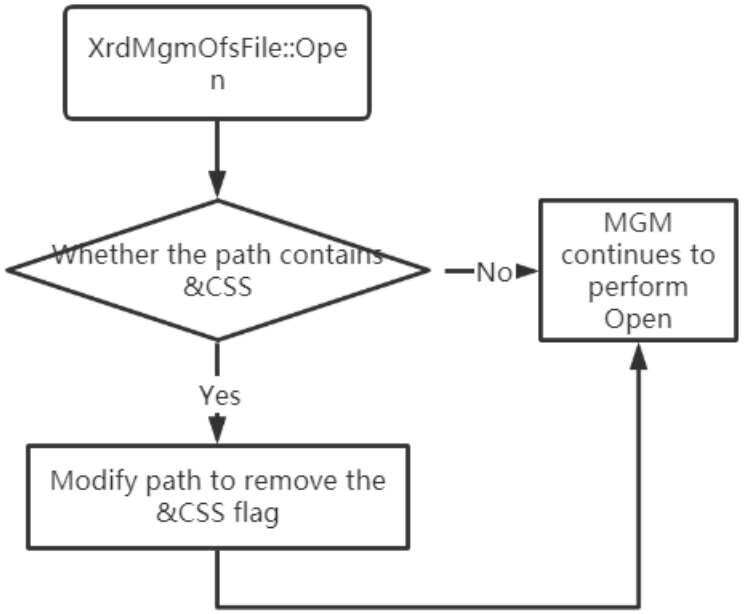


Implementation

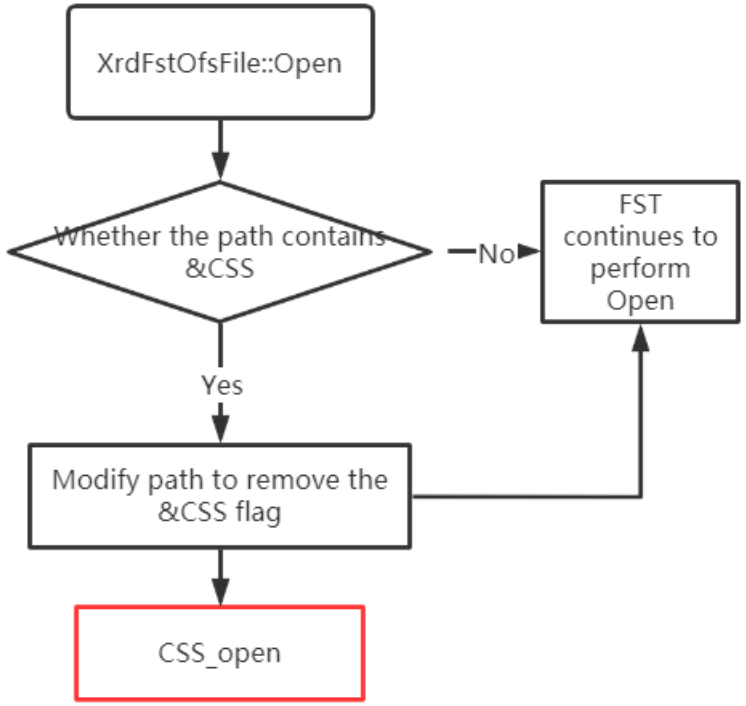
- **Based on the EOS file opening process, we modified MGM and FST respectively.**
- **CSS.MGM**: When the “&CSS” flag appears in the file path passed by the client, restore the file name to the file name normally accessed, and continue to provide the entire modified information to the MGM.
- **CSS.FST**: When the “&CSS” flag appears in the file path passed by the client, CSS_Open is called to start the computable storage service. At the same time, the file path is modified, and the whole modified information is provided to FST to perform normal Open.



Implementation



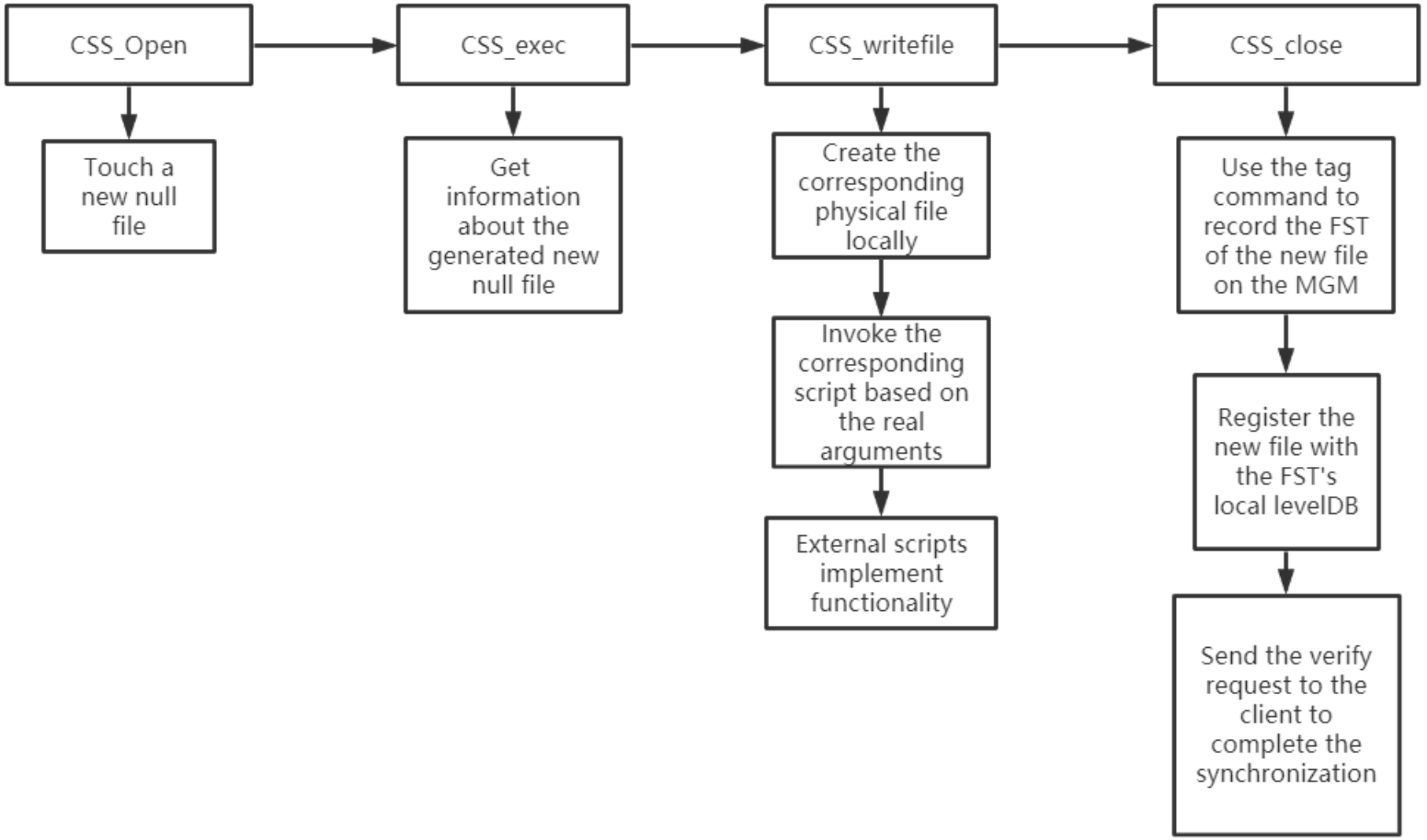
CSS.MGM



CSS.FST

Implementation

CSS.FST：



Implementation

- The main functionality of the CSS described above is encapsulated in the XrdCssOfsFile class.
- To deploy flexible and independent updates, we constructed a link library libEosFstCss.so
- Create EOS/CSS/CMakelists.txt
  - Create an EosFstCss module
- Then modify EOS/FST/CMakelists.txt
  - add_subdirectory(CSS)
  - Add in target_link_libraries of XrdEosFst and XrdEosFst-static
    - EosFstCss

Results

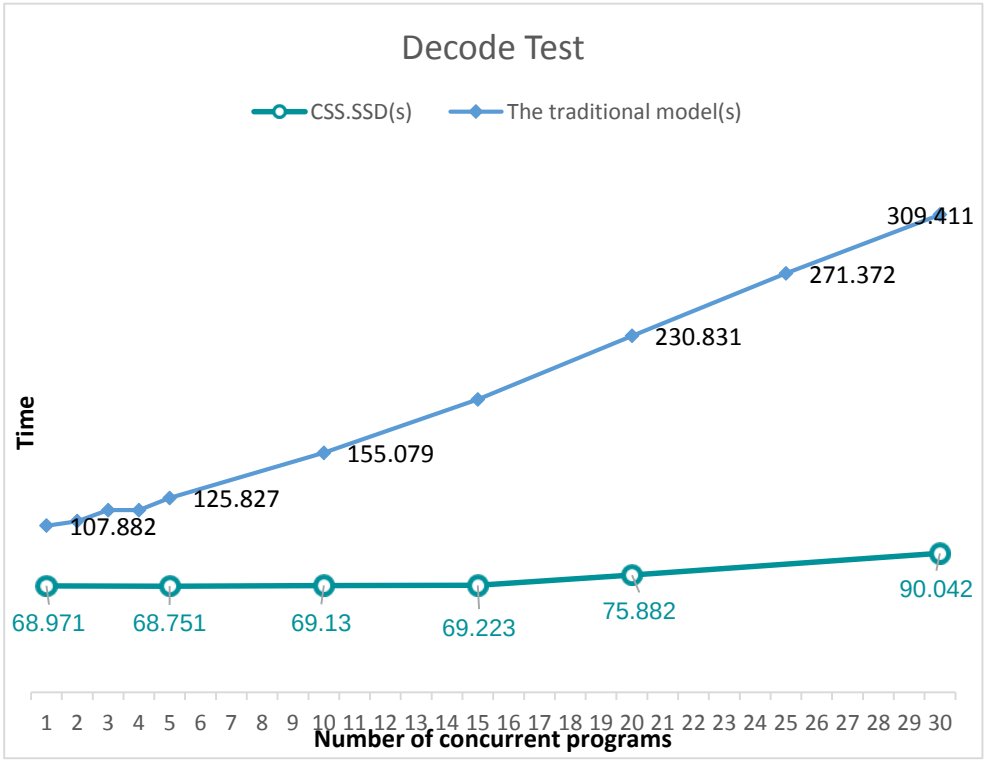
CPU	32 Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.10GHz
Network Interface Card	1Gbps
Storage device	SSD(SATA 3.2,6.0Gb/s) HDD(SATA 3.2,2.1Gb/s)
EOS version	4.7.7
Raw Data File Size	953.7MB

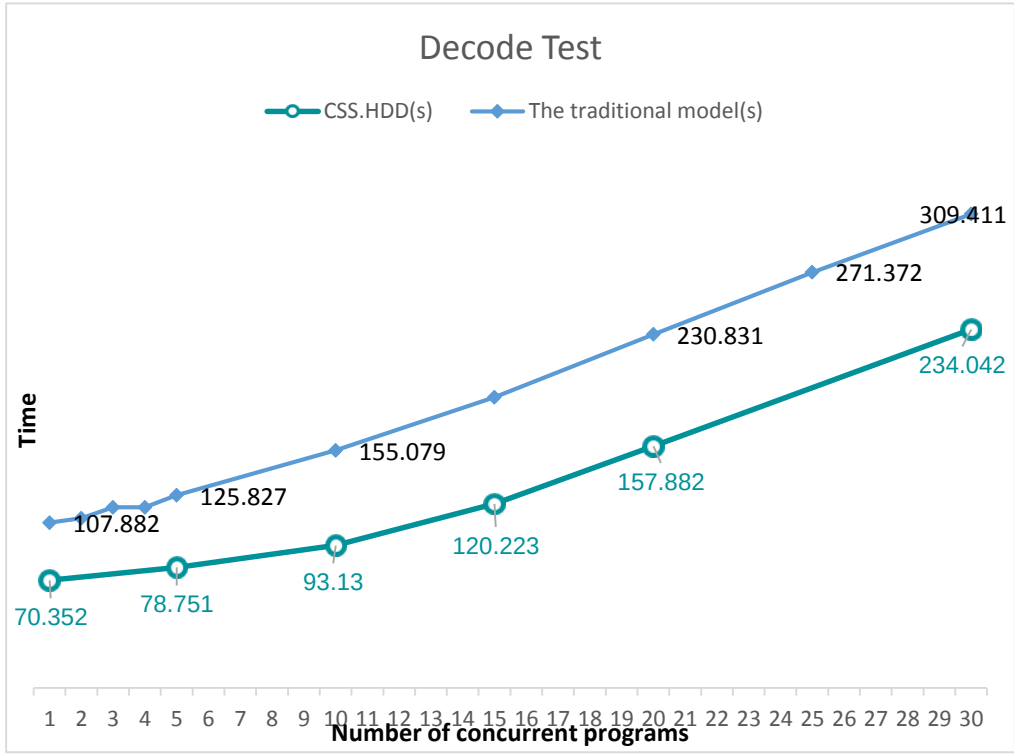
- Decode:Computational functions for testing,I/O intensive tasks

- We tested the decode of the raw data using traditional mode via remote communication and the decode of the raw data using CSS.

Results

In the figure below, the abscissa is the number of parallel programs, and the ordinate is the final time of completion of all programs.





The lower the final time of program operation is, the higher the computational efficiency of the mode is. The smaller the change of the final time of program operation is, the stronger the parallel ability of the mode is.

Conclusion and Outlook

- Computational storage architectures have a good acceleration effect for I/O intensive computations and can increase the amount of parallel tasks.

- But we still need to modify the EOS source code.The next step is to fully encapsulate CSS functionality as plug-ins, enabling plug-and-play.
- The types of CSS provided by extensions are also goals for future implementations.

Thank You