Dynamic I/O congestion control in scalable Lustre file system

Yingjin Qian [1], Ruihai Yi[1]
Yimo Du[2], Nong Xiao[2], Shiyao Jin[2]

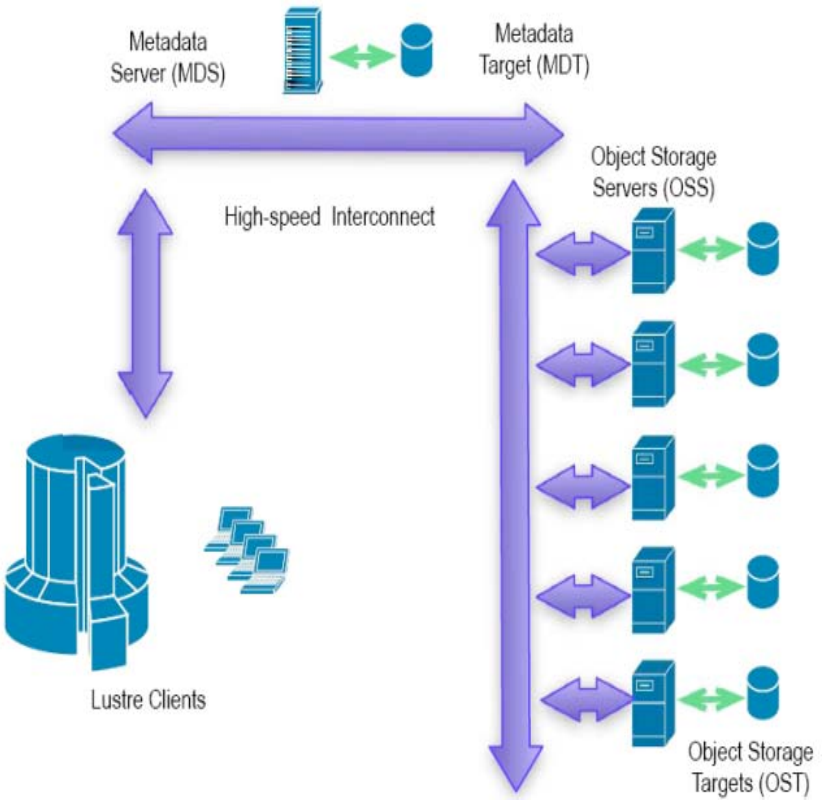
[1]Satellite Marine Tracking & Control Department of China
[2]State key Laboratory of High Performance Computing, National University of Defense Technology

HPC and Lustre

- Scalability is important and challenging for data storage systems.
- Storage cluster is an effective method for storage scalability .
- Lustre has good scalability.
- Lustre is the leading clustered file system in the HPC market.

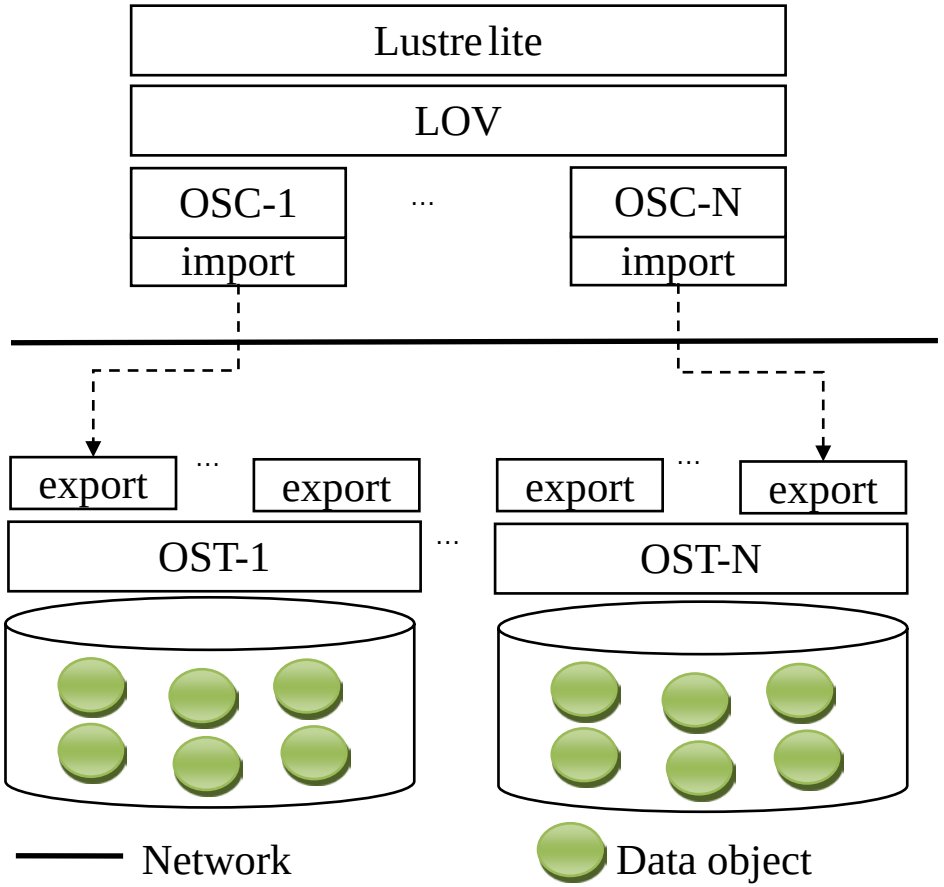
Lustre File System Architecture

- Three components
  - Metadata servers(MDS)
  - object storage servers (OSS)
  - Clients
- One MDS per file system manages one metadata target (MDT)
- OSTs store file data objects



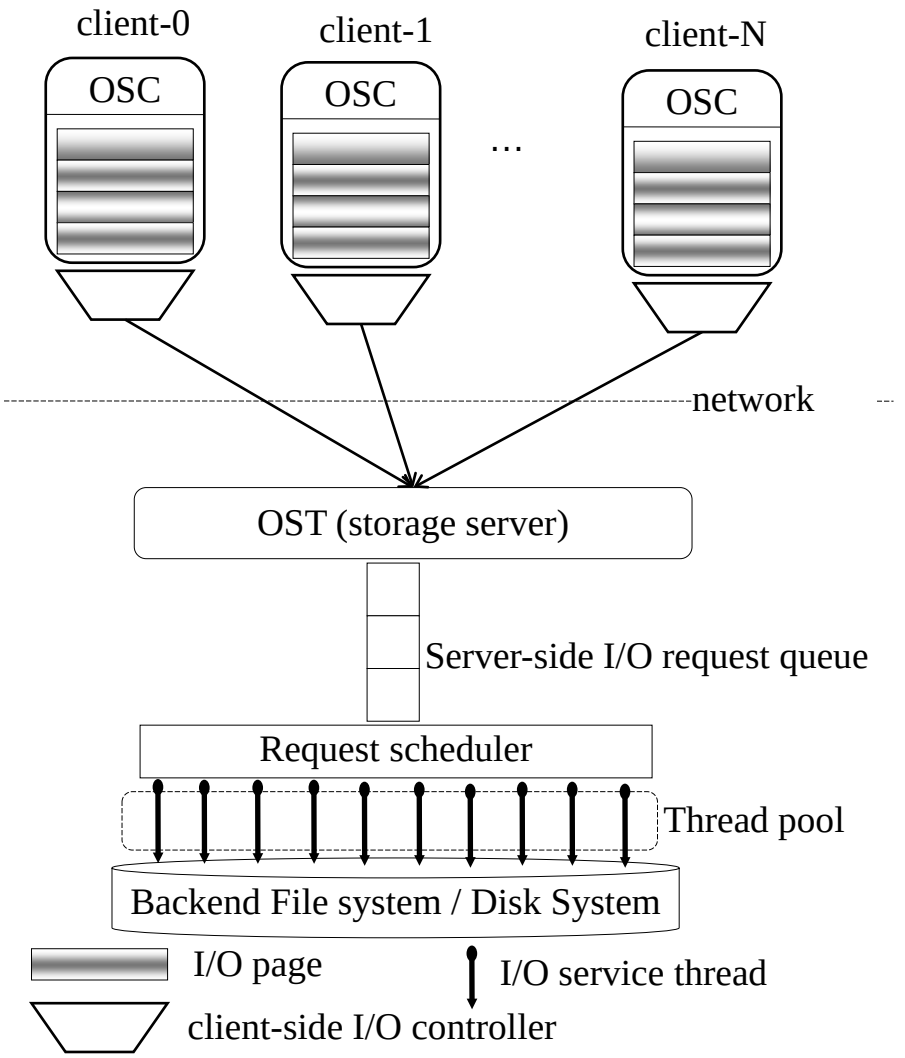
Lustre I/O system

- Lustre lite
  - Hooks with VFS to present VFS interfaces
- Logical Object Volume (LOV)
  - Functions as a virtual object based device for the upper layer stacked upon OSCs
- import/export pair
  - Manage the stateful connections and setup up the communication channel between a client and a server.
- Distributed Lock Manager (DLM)
  - Support fine-grained locking for efficient concurrent file access.



Lustre Scalable I/O model

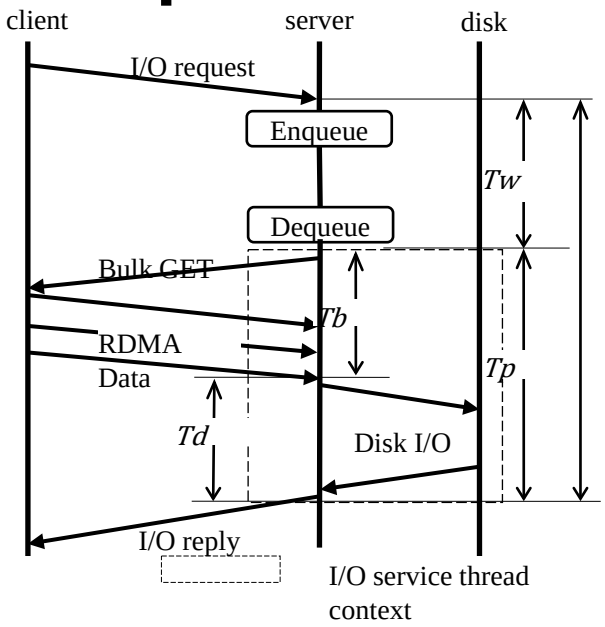
- Large requests are common
- Each client-side import has an I/O controller
- Each client is given an I/O service quota: request concurrency credits(RCC).

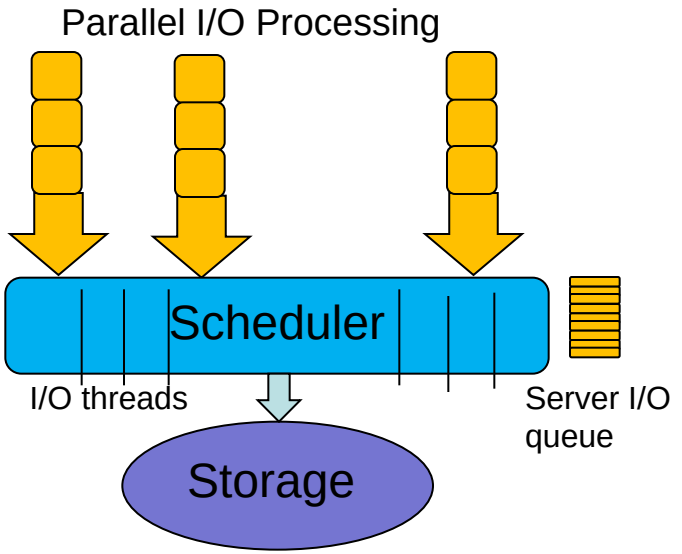


Process I/O request in parallel

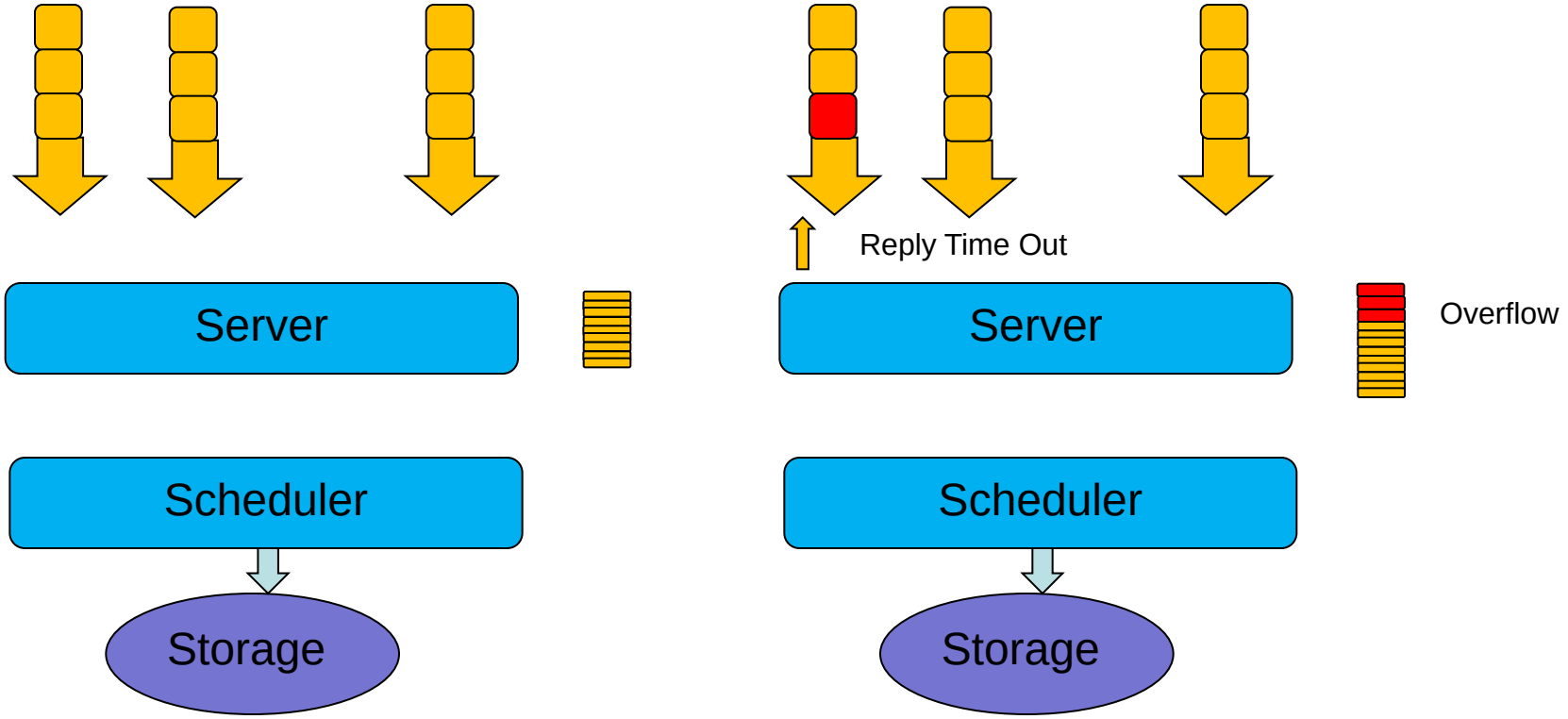
- A write RPC is showing as the right figure.
- Command requests are separated from data requests.
- Request to the server can be processed in parallel.
- All requests from different clients to the backend storage can be scheduled according to the elevator algorithm.

- Compared to traditional network file systems, Lustre is more scalable.





Congestive collapse

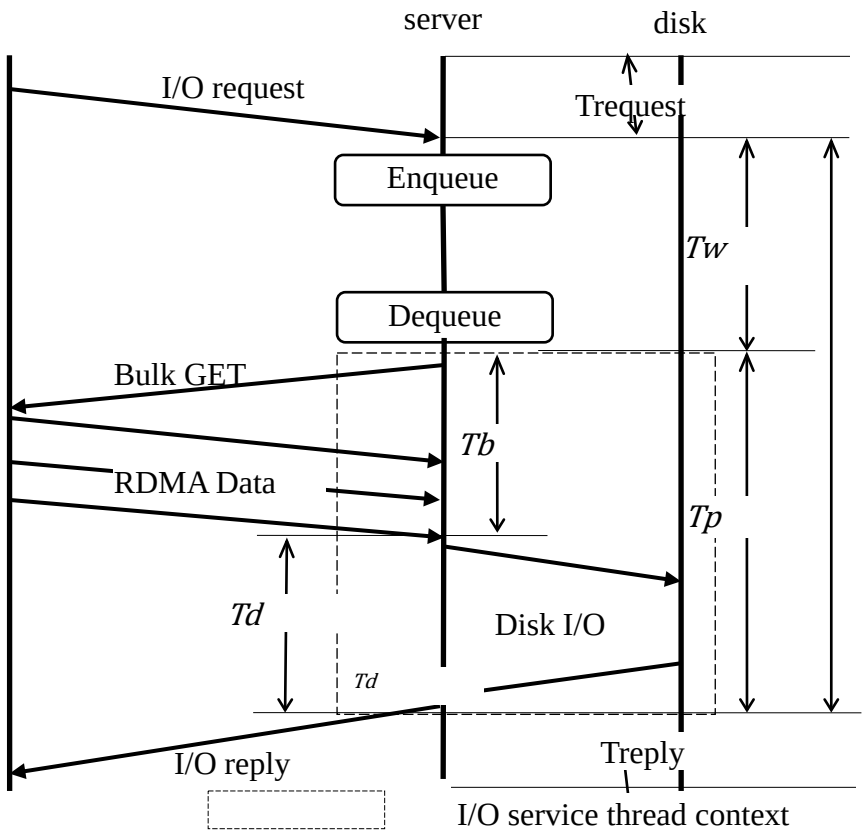


- Storage is slow.
- RPC queue is long.
- Timeout value is set not enough.

What is solutions?

- Set time out value longer
  - Cray Jaguar System: 600 seconds
  - Cons
    - Causes failure detection mechanism to be less responsive
    - Increases the recovery and failover time.
- Solution
  - Limit the number of outstanding I/O requests on the server
- Goal
  - Control the latency not exceeding the timeout value
  - Prevent the occurrence of congestive collapse

Analytic I/O model



$$L = T_n + T_w + T_b + T_d$$

$$= T_n + T_b + \frac{Q+N}{IOPS}$$

T_n, is ignorable compared to other parts

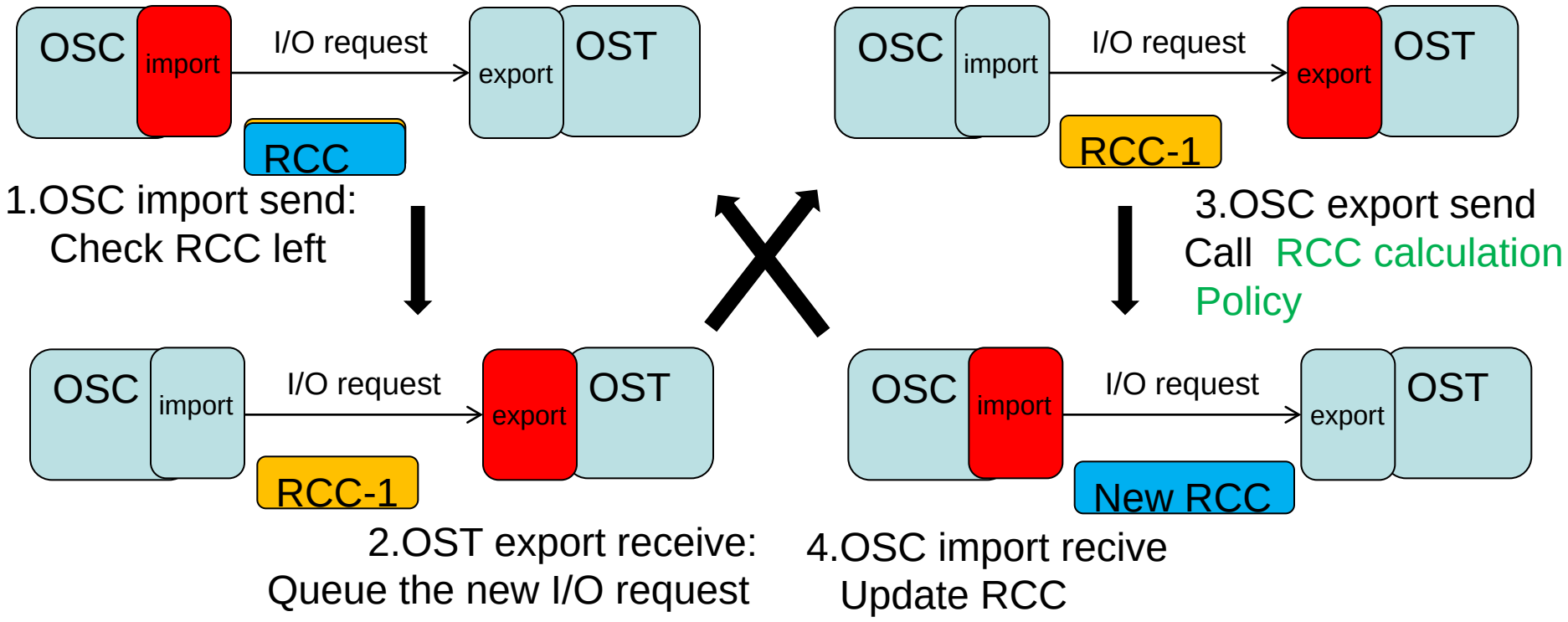
For static RCC scheme

$$L = \frac{(Q+N)}{IOPS} = \frac{D}{IOPS} = \frac{RCC \times C}{IOPS}$$

N	Maximal number of I/O service threads
Q	Queue depth on the server
C	C: The number of I/O active clients

Simple static RCC can not achieve good control effect

Dynamic I/O congestion control



RCC Calculation Policy

- Big RCC
  - Increase locality of disk access pattern
  - For one I/O, its latency can be excessive
- Small RCC
  - Decrease I/O latency for one I/O
  - Impair the locality of disk access pattern
- Goal
  - Within latency bound
  - Enlarge RCC as big as possible
  - Avoid congestive collapse based on length of server queue

Implementation

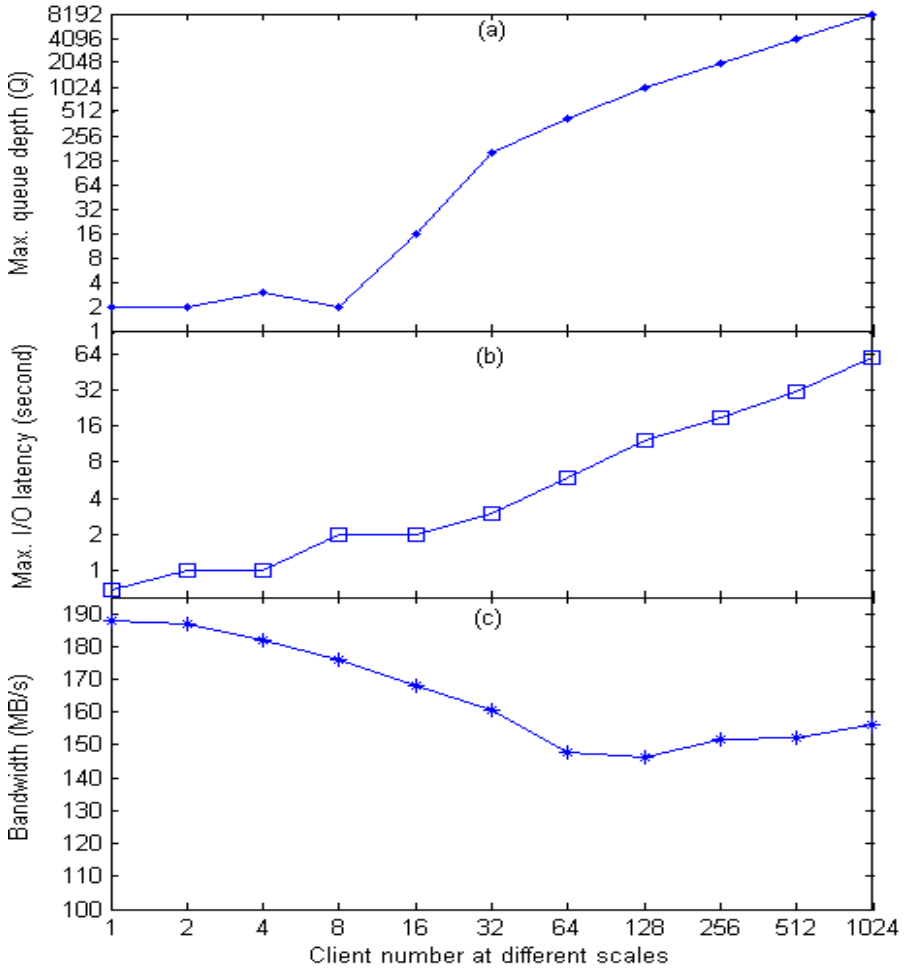
- Implemented our congestion control on the Tianhe-1 supercomputer system
- Configuration of Tianhe-1 and corresponding Lustre file system
  - Scale: 80 compute cabinets including 2560 compute nodes+512 operation nodes;
  - Node configuration: 2 AMD GPUs+2 Intel Xeon processor; 32GB memory;
  - Interconnect: Mellanox Infiniband switches 40Gb/s
  - Storage server: Nehlem CPU; 8GB memory; linux kernel with Lustre-specific patches.; single OST bandwith 206MB/s
  - Storage Space Capacity: 1PB

Experiment Setup

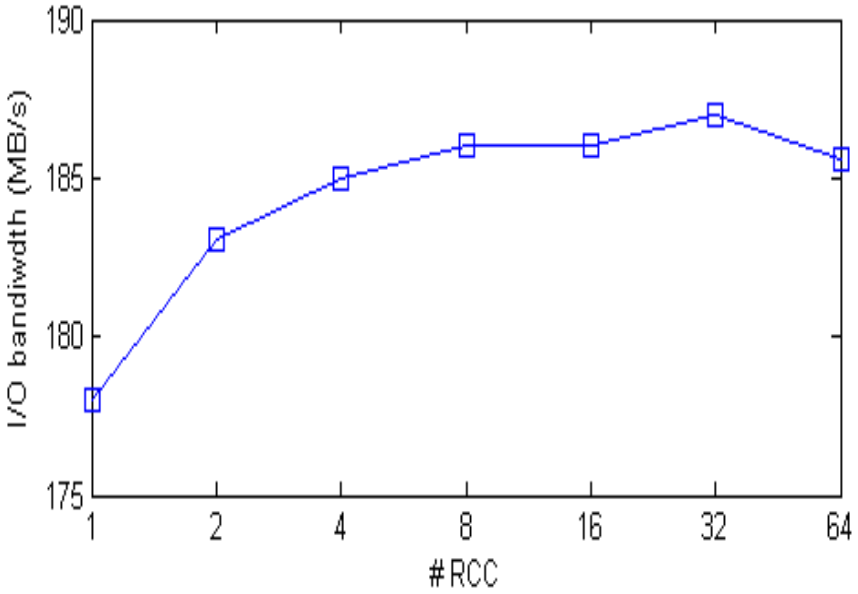
- Tianhe-1 Lustre configuration
  - Clients scale: 1 to 1024
  - Synthetic workload: IOR benchmark
- Workload characterization
  - File size: Fixed 512GB
  - I/O size: 512G/c per client in average
  - I/O mode: File per Processor (FPP)
  - Transfer size: 1MB
  - Parameters by default:

RCC_{min}	1	STL	60s
RCC_{max}	32	CTL	0
D_{low}	128		

Base Line Evaluation



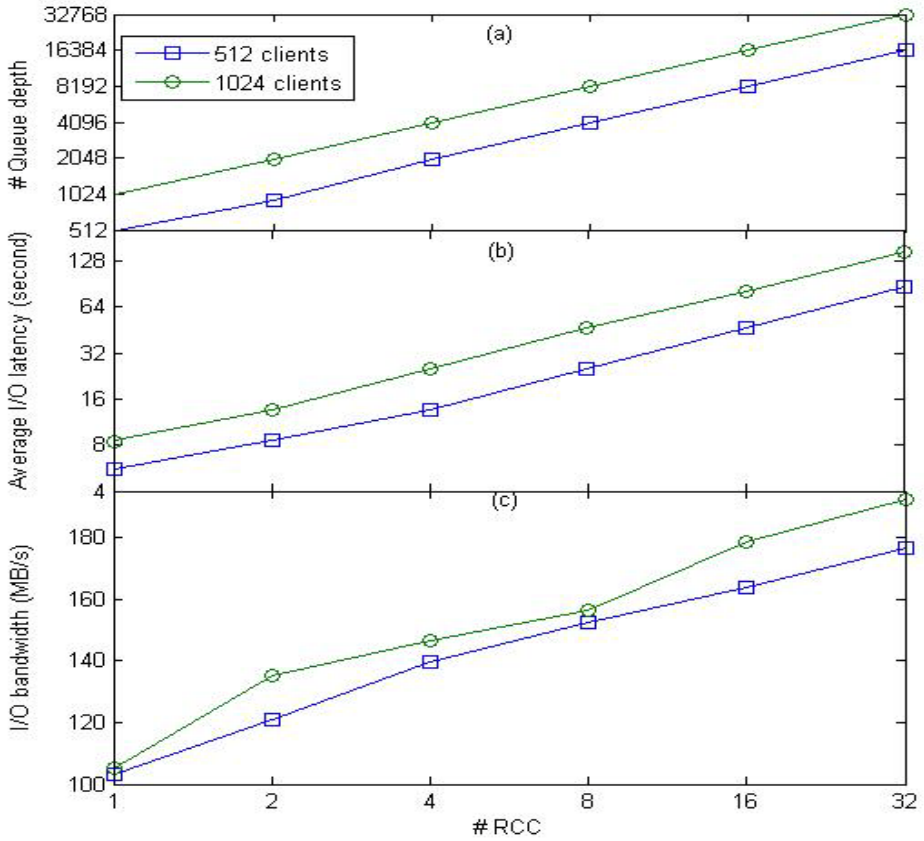
static RCC scheme at scale (n=8)



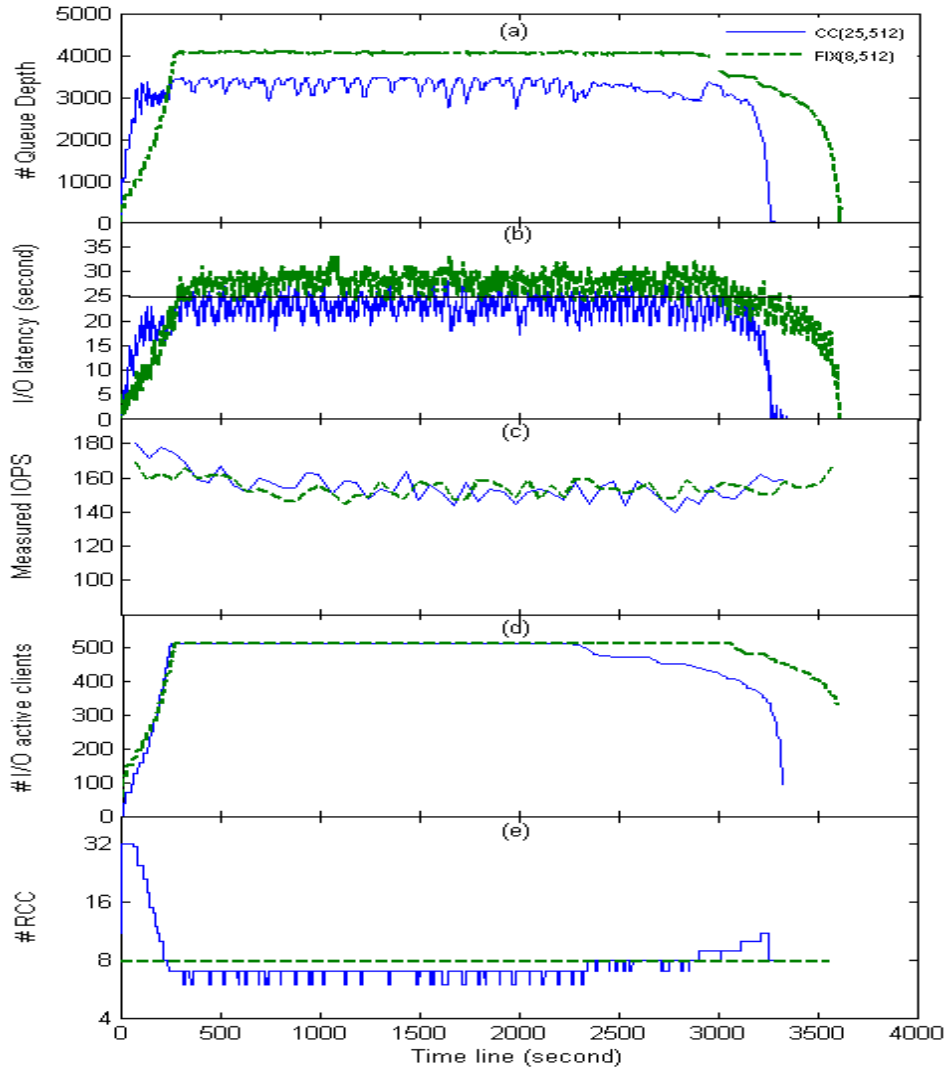
static RCC scheme with different n values (c=1)

Base Line Evaluation

Performance of the static RCC scheme with different *RCC* values at large scale

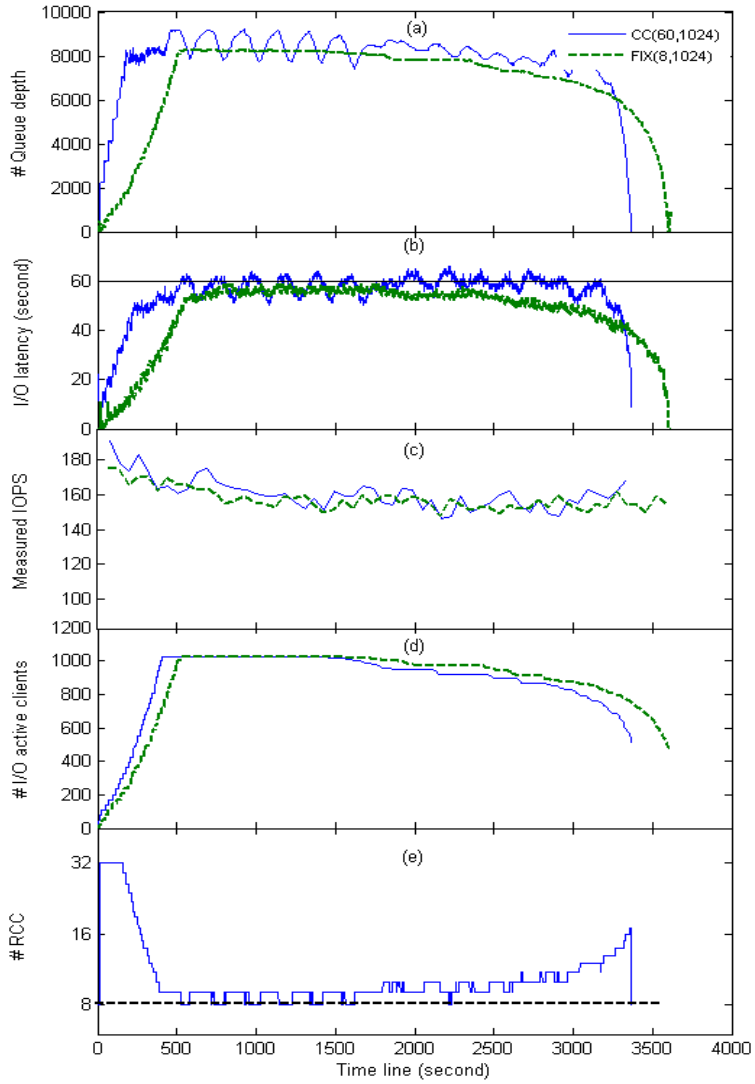


Dynamic RCC Evaluation



Comparison between static RCC and dynamic RCC with 512 clients

Dynamic RCC Evaluation



Comparison between static RCC and dynamic RCC with 1024 clients

Any Questions?

- Thank you!