CSE 3320
Operating Systems
Distributed and Parallel File Systems

Jia Rao

Department of Computer Science and Engineering

http://ranger.uta.edu/~jrao

Recap of Previous Classes

- File systems provide an abstraction of permanently stored data
  - Namespace: files and directories
    - Translate paths to corresponding locations on disks
  - Space management and optimizations
    - Free blocks
    - Caching and prefetching
  - Reliability and consistency

Distributed and Parallel File Systems

- Provide similar abstractions of data on *multiple* machines
  - Namespace: path name → machine ID: disk block address
  - Management: placement of files on machines
    - Replication
    - Striping
- Designed for performance and availability

Distributed v.s. Parallel File Systems

- Design objectives

  The boundary is blurring

  - Fault-tolerance v.s. Concurrent performance
- Data distribution
  - Entire file on a single node v.s. striping over multi nodes
- Symmetry
  - Storage co-located with apps v.s. storage separated from apps
- Fault-tolerance
  - Designed for fault-tolerance v.s. relying on enterprise storage
- Workload
  - Loosely coupled, distributed apps v.s. coordinated HPC apps

Examples

- Distributed File Systems
  - NFS, GFS (Google File System), HDFS (Hadoop Distributed File System), GlusterFS
- Parallel File Systems
  - PVFS (Parallel Virtual File System), Lustre, OCFS2, GPFS

Design Issues (1)

- Nameserver
  - maps file names to objects (files, directories, blocks)
  - Implementation options
    - Single name Server
      - Simple implementation, reliability and performance issues
    - Several Name Servers (on different hosts)
      - Each server responsible for a domain

Design Issues (2)

- Caching
  - Caching at the client: Main memory vs. Disk
  - Cache consistency
    - Server initiated
      - Server informs cache managers when data in client caches is stale
      - Client cache managers invalidate stale data or retrieve new data
      - Disadvantage: extensive communication
    - Client initiated
      - Cache managers at the clients validate data with server before returning it to clients
      - Disadvantage: extensive communication
    - Prohibit file caching when concurrent-writing
      - Several clients open a file, at least one of them for writing
      - Server informs all clients to purge that cached file
    - Lock files when concurrent-write sharing (at least one client opens for write)

Design Issues (3)

- Update (write) policy
  - Once a client writes into a file (and the local cache), when should the modified cache be sent to the server?
    - Write-through: all writes at the clients, immediately transferred to the servers
      - Advantage: reliability
      - Disadvantage: performance, it does not take advantage of the cache
    - Delayed writing: delay transfer to servers
      - Advantages:
        - Many writes take place (including intermediate results) before a transfer
        - Some data may be lost
      - Disadvantage: reliability
    - Delayed writing until file is closed at client
      - For short open intervals, same as delayed writing
      - For long intervals, reliability problems

Design Issues (4)

Availability

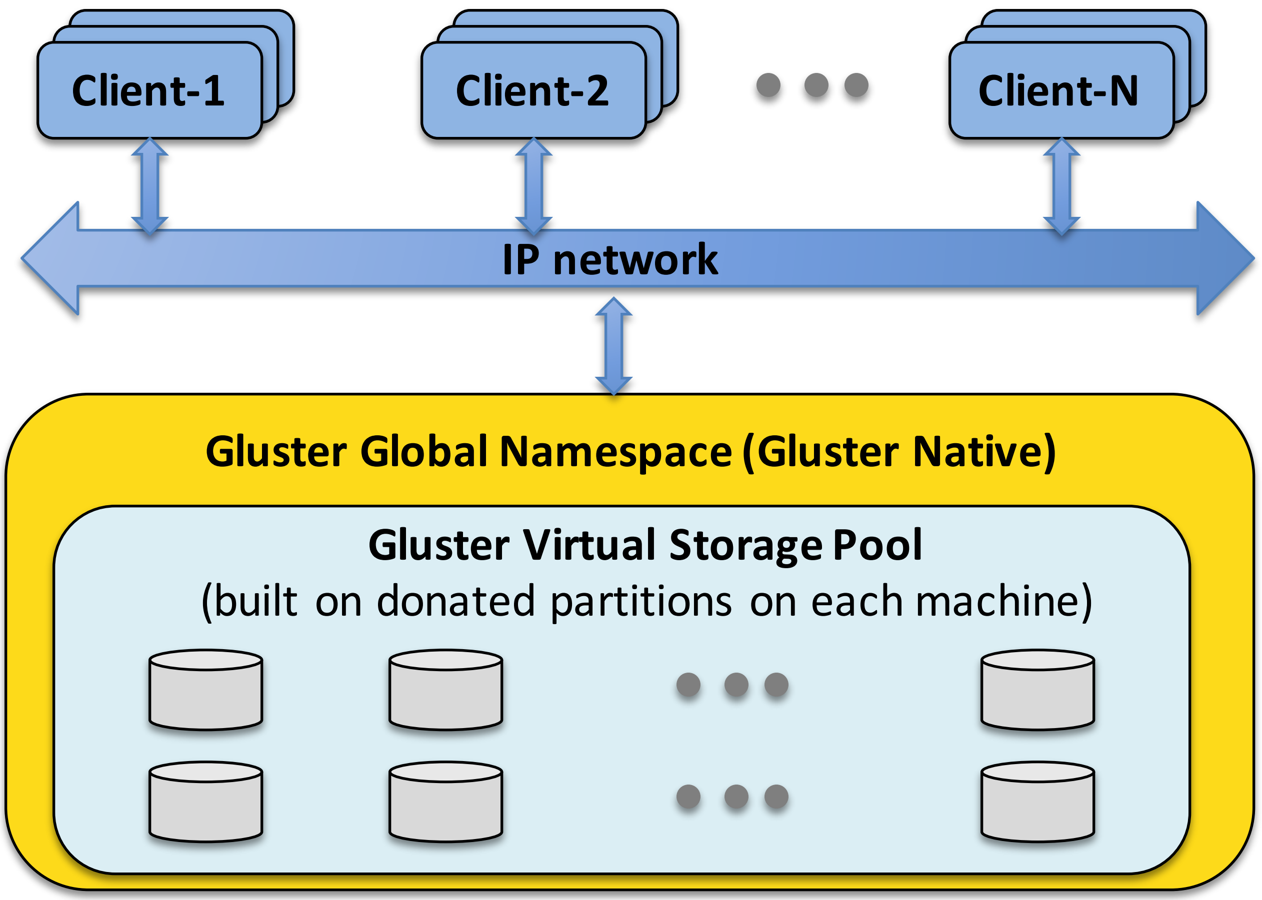
- What is the level of availability of files in a distributed file system?
- Use replication to increase availability, i.e. many copies (replicas) of files are maintained at different sites/servers
- Replication issues:
  - How to keep replicas consistent
  - How to detect inconsistency among replicas

Design Issues (5)

Scalability

- Deal with a growing system?
- Issues
  - Node join and leave (fail)
  - Cache consistency
  - Nameserver
- Solutions
  - Replication
  - Design cache consistency protocol for scalability
  - Multiple name (meta) servers
  - Take advantage of multi-thread and multi-core

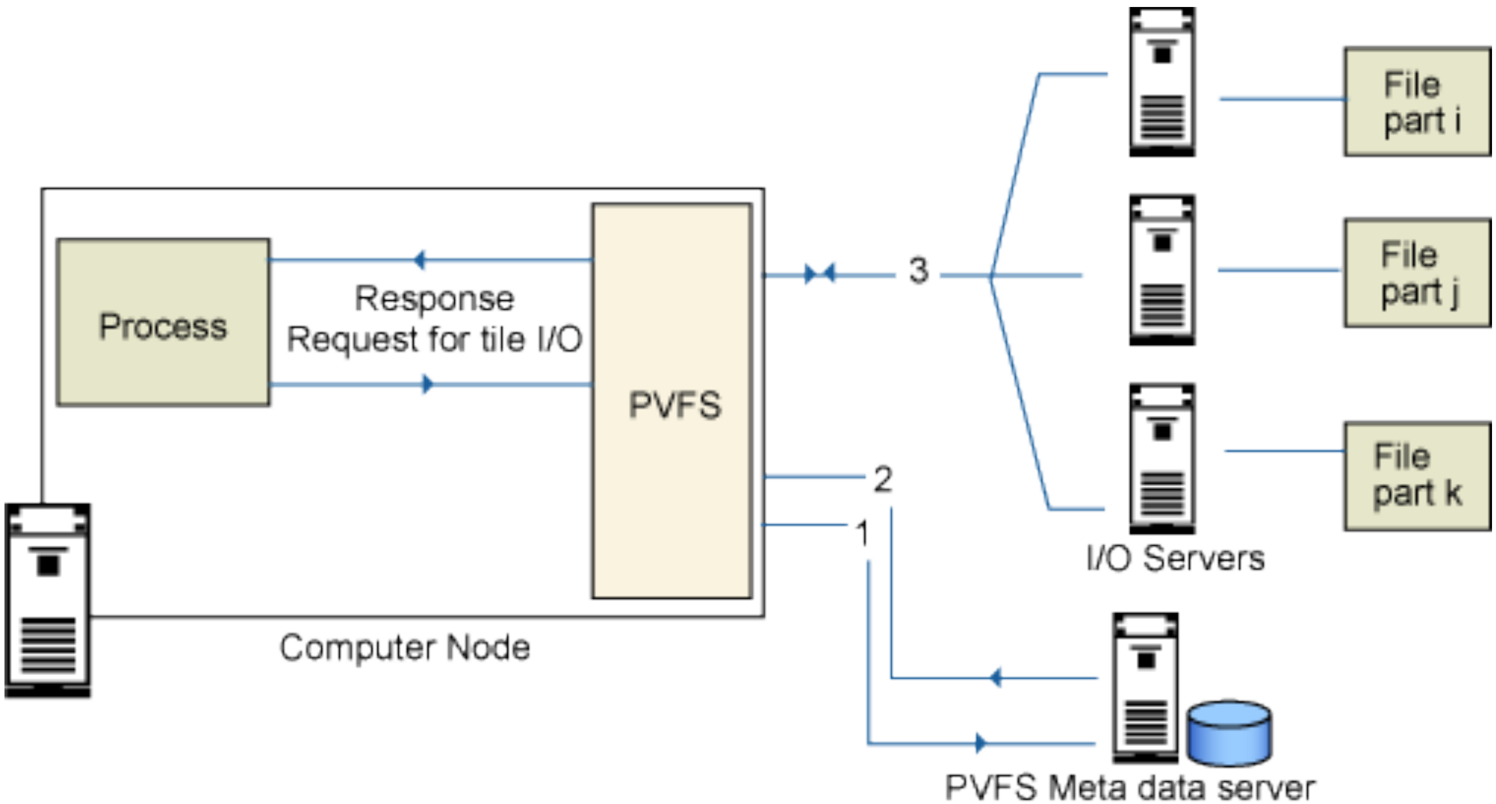
Example - GlusterFS (DFS)



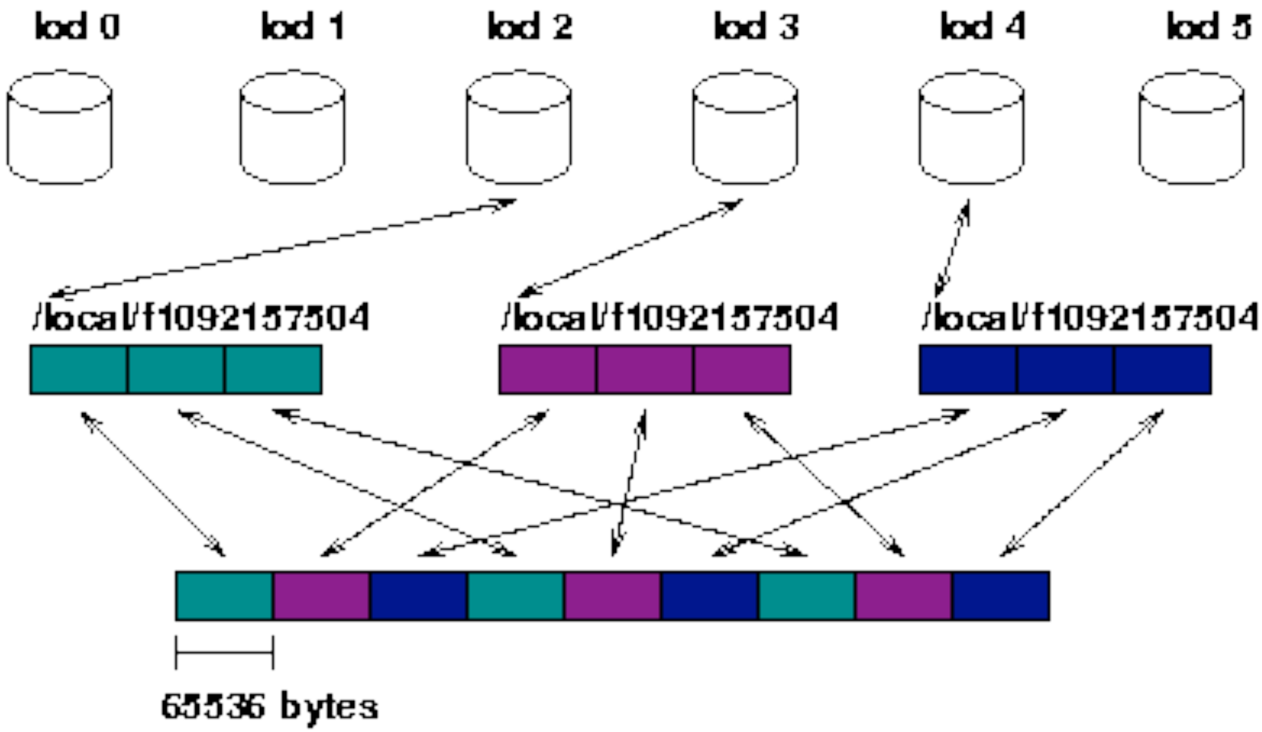
Example – GlusterFS (2)

- Three ways to place files
  - Distribute: place entire files on different servers
    - Pros: good scalability, efficient disk space usage
    - Cons: poor reliability
  - Replicate: place identical copies of files on different servers
    - Pros: reliability
    - Cons: wasted disk space, moderate scalability
  - Stripe: place only part of a file on one server
    - Pros: good performance for concurrent and random access
    - Cons: poor scalability and reliability

Example – PVFS(PFS)



Example – PVFS (PFS)



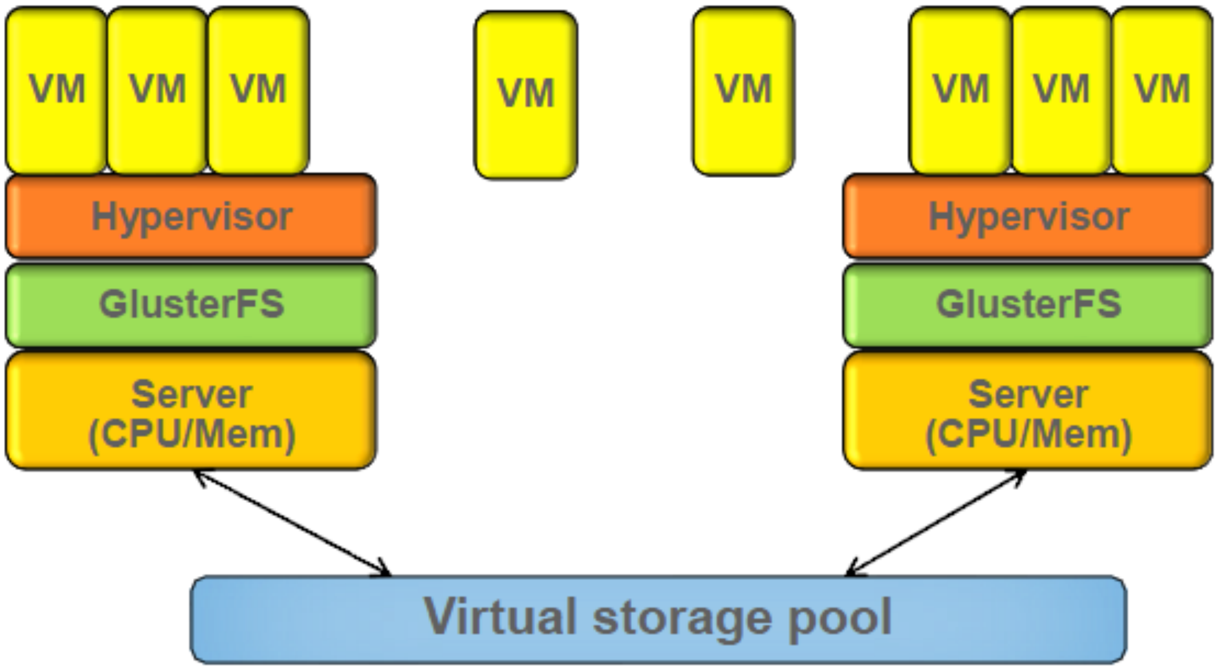
Significant improvement in throughput

What could be the issues?

1. Sever coordination affects efficiency
2. Client QoS?

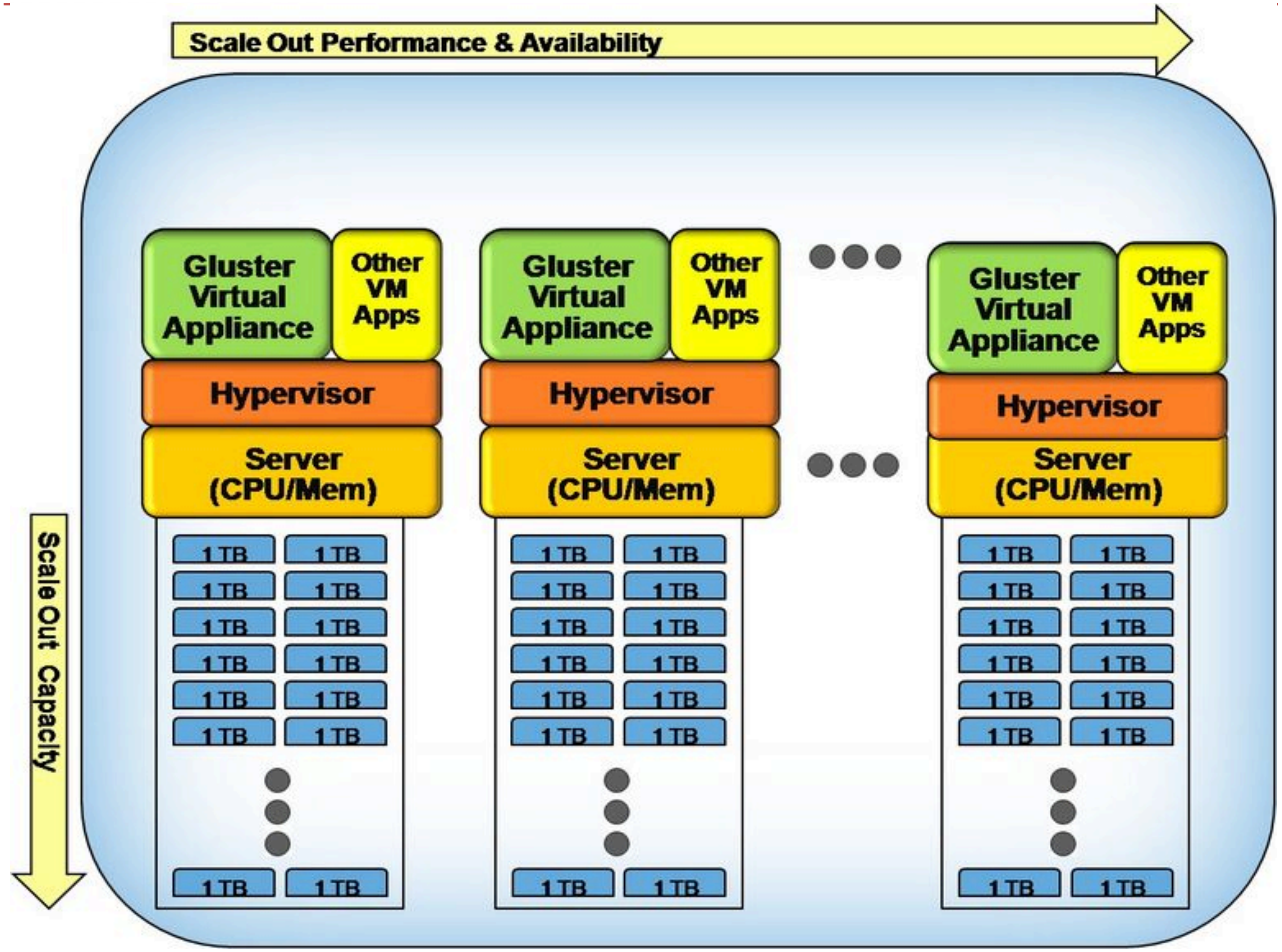
DFS and PFS in the Cloud (1)

- Both approaches provide cheap, reliable and high-performance cloud storage solutions



Use case-1

DFS and PFS in the Cloud (2)



Use case-2

Some Real Results…

- Host a 8-VM Hadoop cluster on 8 DELL machines
- Performed micro and real I/O intensive workloads
- Two storage solutions: PVFS and local ext3

	PVFS	Local ext3
Gridmix websort 20GB data	2391 second	4693 second

PVFS

	16k	32k	64k	256k	1M
Sequential	58.89	60.15	60.47	104.80	130.47
random	12.34	20.84	33.51	50.43	108.71

Network bandwidth bottleneck

Local ext3

	16k	32k	64k	256k	1M
Sequential	120.11	120.56	120.39	120.39	120.57
random	4.01	7.80	14.71	43.20	92.19