Cluster Computing

Parallel I/O for Clusters

- Introduction
- A Case for Cluster I/O Systems
- The Parallel I/O Problem
- File Abstraction
- Methods and Techniques
- Architectures and Systems
- The ViPIOS Approach
- Conclusions and Future Trends

2

Introduction

- Microprocessors increase their performance by 50% to 100% per year, but disk hardly their performance
  - more exaggerated in processing huge data sets
    - smart parallel I/O algorithms are required to access the data on disk efficiently
- In high performance computing the application shifted from being CPU-bound to be I/O bound
- Performance cannot be scaled up by increasing the number of CPUs any more, but by increasing the bandwidth of the I/O subsystem

3

Introduction

- Cluster-based parallel I/O systems
- ViPIOS
  - client-server based tool
    - combining capabilities found in parallel I/O runtime libraries and parallel file systems

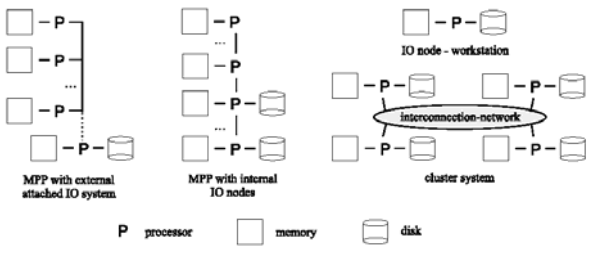
4

A Case for Cluster I/O Systems

- An architectural framework similar to a cluster of computers can be found in state-of-the-art MPP systems
- Two different types of I/O systems for MPPs
  - external attached I/O subsystems
    - connected with the MPP by a separate interconnection bus
    - dedicated bus
    - system administration and application development are relatively simple
  - internal I/O nodes
    - an I/O node is a conventional processing node with a disk attached
    - only the software distinguishes between I/O and computing node

5

I/O Architecture Topologies



6

A Case for Cluster I/O Systems

- MPP I/O framework is similar to cluster I/O framework
- Eminent drawback is the latency of the interconnection network
  - Computational-bound application
    - It is overcome by applying new bus technologies (Giganet, Myrinet) or by using multiple I/O buses in parallel
  - I/O based application
- The already developed methods and paradigms to overcome this I/O bottleneck on MPP systems can similarly be used on cluster system.
  - Clusters are a suitable platform for I/O based applications

7

The Parallel I/O Problem

- IO needs of typical supercomputing applications
  - Input
  - Debugging
  - Scratch file
  - Checkpoint/restart
  - Output
  - Accessing out-of-core structure
- Regular Problem
  - This approach is supported by the well-known SPMD model
    - Single thread control-parallel execution
    - Global namespace
    - Loose synchronization
  - This model design a sequential program and transfer it to parallel execution

8

The Parallel I/O Problem

- Irregular Problem
  - Data access patterns cannot be predicted until runtime
  - Three different kinds to be optimized
    - Irregular control structures
      - conditional statements making it inefficient to run on synchronous programming models
    - Irregular data structures
      - unbalanced trees or graphs
    - Irregular communication patterns
      - these lead to non-determinism
- Out of Core (OOC) Computation
  - Primary data structure do not wholly fit into main memory
  - Computation is carried out in several phases
    - Data is brought into main memory
    - Process
    - Store back onto secondary storage

9

File Abstraction

- Files can be accessed synchronized
  - By collective file operations
- 4 execution modes for parallel file I/O
  - Broadcast-reduce
    - access the same data collectively
    - for a read, all processes get a copy of the read data
    - for a write, data is written only once
  - Scatter-gather
    - all processes access different parts of the file
  - Shared offset
    - the processes share a common file handle but access the file individually
  - Independent
    - each process has its own private handle

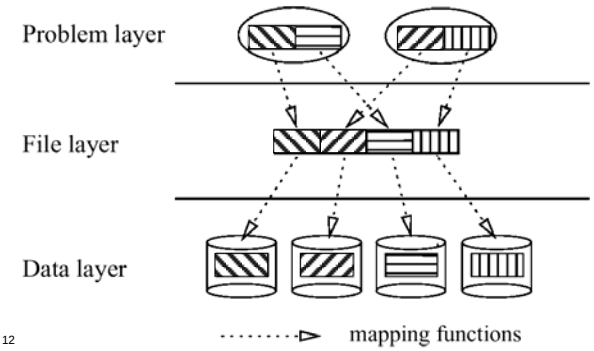
10

File Abstraction

- Layers in the file architecture to allow flexibility for the programmer and the administration methods
- Three independent layers in the file architecture
  - Problem layer
    - Define problem specific data distribution among cooperating parallel processes
  - File layer
    - Provide a composed view of persistently stored data in the system
  - Data layer
    - Define physical data distribution among the available disks

11

File Abstraction



12

File Abstraction

- Mapping functions between these layers
  - Logical data independence
    - exist between the problem and the file layer
  - Physical data independence
    - exist between the file and data layer

13

Methods and Techniques

- Aim to accomplish followings
  - Maximize the use of available parallel I/O device to increase the bandwidth
  - Minimize the number of disk read and write operation per device
  - Minimize the number of I/O specific messages between processes to avoid unnecessary costly communication
  - Maximize hit ratio (ratio between accessed data to requested data): data sieving

14

Methods and Techniques

- Three groups of methods in the parallel I/O execution framework
  - Application level methods
    - Try to organize the main memory objects mapping the disk space to make disk accesses efficient
      - Buffering algorithm
    - By runtime libraries which are linked to the application programs
    - The application program performs the data accesses itself without the need for dedicated I/O server programs
  - I/O level methods
    - Try to reorganize the disk access requests of the application programs to achieve better performance
    - By independent I/O node servers which collect the requests and perform the accesses
  - Access anticipation methods
    - Anticipate data access patterns which are drawn by hints from the code advance to its execution

15

Two Phase Method

- A method for reading/writing in-core arrays from/to disk
- Base on the fact that I/O performance is better when process make an small number of large request instead of a large number of small ones
  - The processes read data from disk
  - Data is redistributed among the processes by inter-process communication
- ETPM
  - Several I/O requests are combined into fewer larger requests
    - eliminate multiple disk accesses for the same data (reducing contention for disks)
  - All processes have to participate in the collective operation
    - Advantage: process can cooperate to perform certain optimizations when the same data is required by more processes

16

Disk Directed I/O

- Improve the performance of reading/writing large, regular data structures such as a matrix distributed between memory and distributed disks
- In traditional Unix-like parallel filesystems, individual processor make a request to the file system for each piece of data
  - If many processes request the same data, a file is read multiple times and the data is transmitted to each processor independently
- Interface that support collective I/O
  - all processes make a single joint request to the file system
  - optimizing data transfer
- High-level request is sent to an I/O node
  - usage of less memory, less CPU and message passing overhead
- Double-buffering
- Special remote-memory "get" and "put" message
  - to pipeline the transfer of data

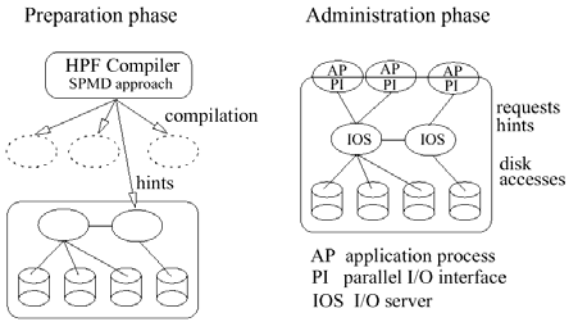
17

Two-Phase Data Administration

- Try to anticipate data access patterns of the application program as early as possible in the program execution cycle
- Management of data is split into two distinct phases
  - Preparation phase
    - data layout decisions, data distribution operations
    - precedes the execution of the application processes
    - uses the information collected during the application program compilation process.
    - physical data layout schemes are defined
    - actual server process for each application process and disks for the stored data are chosen
    - data storage areas are prepared
    - main memory buffers allocated
  - Administration phase
    - data accesses and data prefetching
    - accomplishes the I/O requests of the application processes and performs necessary reorganization of the data layout

18

Two-Phase Data Administration



19

Two-Phase Data Administration

- Hints are the general tool to support the I/O subsystem with information for the data administration process
  - Hints: data and problem-specific information provided by the application programmer or compiler
  - File administration hints
    - provide information on the problem specific data distribution of the application processes
    - high parallelization can be reached if the problem specific data distribution of the application processes matches the physical data layout on disk
  - Data prefetching hints
    - yield better performance by pipelined parallelism and file alignment
  - Administration hints
    - allow the configuration of the I/O subsystem according to the problem situation respective to the underlying hardware characteristics and their specific I/O needs

20

Architectures and Systems

- To solve I/O bottleneck problem
- Two different approaches
  - Parallel file systems
    - low level solution
    - OS is enhanced by special features that deal directly with I/O at the level of files
  - Runtime library for high performance language
    - enhance conventional high performance language such as Fortran or C/C++

21

Runtime Modules and Libraries

- The aim is to adapt graciously to the requirements of the problem characteristics specified in the application program and tools for the application programmer
- The executing application can hardly react dynamically to changing system situations or problem characteristics
  - the data decisions were made during the programming and not during the execution phase
- The CPU of a node has to accomplish both the application processing and the I/O request of the application
- The optimal performance is nearly impossible to reach by the usage of runtime libraries

22

I/O Runtime Libraries

name	institution	sync/async	SPMD/SIMD/MIMD	strided access	data parallel	message passing	client-server	clustering	caching	prefetching	collective operations	shared file pointer	concurrency control	views	new ideas
ADIO	Dartmouth College	+/+	+/-/+	+	+	+	-	+	-	-	+	-	+	+	strategies for implementing APIs
CVL	Dartmouth College		/+/												vector operations
DDLY	University of Malaga		/+/			+					+				VDS, IDS
Jovian	University of Maryland	+/	+/ /			+				+	+			+	global, distributed view
MPI-2	MPI Forum	+/+	+/+/+/+	+		+					+	+		+	I/O for MPI
MPI-IO	MPI-IO Committee	+/+	+/+/+			+					+	+		+	I/O for MPI
Multipol	University of California	+/+								-					PD, distributed data structures
Panda	University of Illinois	+/+	+/+/+	+	+	+	+			+	+	-	+	+	server-directed I/O, chunking
PASSION	University of Syracuse	+/	+/ /		+	+				+	+	+			TPM, irregular problems
ROMIO	Darmouth College	+/+	+/-/+	+	+	+	+	-	-	-	-	-	+	+	portable implementation of MPI-IO
TPIE	Duke Uni., Uni. of Delware														
ViPIOS	University of Vienna	+/+	+/+/	+	+	+	+	+	+	+	+	+		+	influence from DB technology

Annotation: + ... "is supported"
- ... "is not supported"

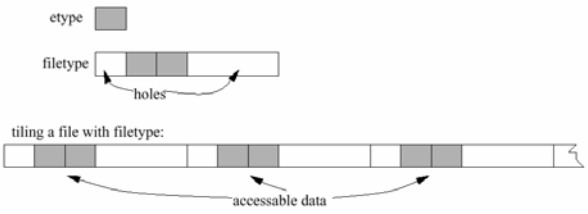
23

MPI-I/O

- Parallel I/O interface based on MPI message passing framework
- Supports a high-level interface
  - partition files among multiple processes
  - transfer global data structures between process memories and files
  - optimizations of physical file layout on storage device
- Three access functions
  - Positioning is accomplished by explicit offsets, individual file pointers, and shared file pointers
  - Synchronization and asynchronization (blocking, nonblocking, respectively)
  - Coordination by collective operations
- Typical implementation of MPI-IO
  - PMPIO: Portable MPI I/O by NASA Ames Research Center
  - ROMIO: as MPI-IO extension of MPICH by Argonne National Lab.
  - MPI-IO/PIOFS: by IBM Watson Research Center
  - HPSS implementation: by Lawrence Livermore National Lab.

24

Tiling a File Using a Filetype



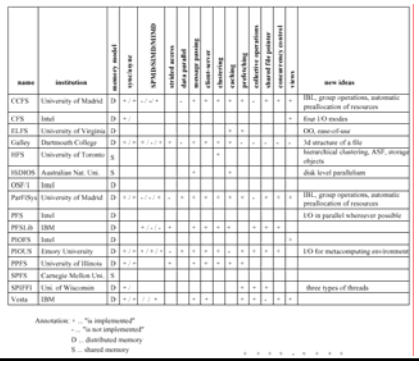
25

Parallel File Systems

- Parallel disk access via a parallel filesystem interface
- Balance the parallel processing capabilities of their processor architectures with the I/O capabilities of a parallel I/O subsystem
- Compared to runtime libraries, parallel filesystems have the advantage that they execute independently from their application
  - execute independently from their application
  - I/O servers are directly support
  - Debugging
  - Tracing
  - Checkpointing
- Drawback
  - not support the capabilities of the available high performance languages directly
  - disallow the programmer to coordinate the disk access according to the distribution profile of the problem specification

26

Parallel File Systems



name	institution	memory model	structured	SPMD/MIMD/MPMD	strided access	data parallel	message passing	client-server	clustering	caching	prefetching	collective operations	shared file pointer	concurrency control	views	new ideas
CCFS	University of Madrid	D	+/-/+	-/-/+		-	+	+	+	+	+	-	+	+	+	IBL, group operations, automatic preallocation of resources
CFS	Intel	D	+/-												+	four I/O modes
ELFS	University of Virginia	D								+	+					OO, ease-of-use
Galley	Dartmouth College	D	+/+	+/+/+	+	-	+	+	+	+	-	-	-	-	-	M structure of a file
HFS	University of Toronto	S						+	+							hierarchical clustering, ASF, storage objects
HiDIOS	Australian Nat. Uni.	S					+		+							disk level parallelism
OSF/1	Intel	D														
ParFSys	University of Madrid	D	+/+	-/-/+	+	+	+	+	+	+	+	+	+	+	+	IBL, group operations, automatic preallocation of resources
PFS	Intel	D														I/O in parallel wherever possible
PFSLib	IBM	D		+/-/-	+		+	+	+			+	+	+		
PIOFS	Intel	D													+	
PIOUS	Emory University	D	+/+	+/+/-	+	+	+	+	+	+	+	+	+	+		I/O for metacomputing environment
PPFS	University of Illinois	D	+/+		+		+	+	+	+						
SPFS	Carnegie Mellon Uni.	S														
SPIFFI	Uni. of Wisconsin	D	+/-									+	+			three types of threads
Vesta	IBM	D	+/-/+	/	+		+	+		+	+	-	+	+		

Annotation: + ... "is implemented"
- ... "is not implemented"
D ... distributed memory
S ... shared memory

27

Parallel Database Systems

- Link to the spread of the relational model as the common user interface
- Relational queries are well suited for parallel execution
- Two different types of parallelism can be distinguished
  - Inter-operator parallelism
    - Parallelism from independent, but simultaneous, execution of different operators of the query execution tree
  - Intra-operator parallelism
    - Partitioning input data stream of one operator to a number of parallel query processes
    - Each process execute the same operation on a different part of data

28

The ViPIOS Approach

- By the University of Vienna
- A full-fledged parallel I/O runtime system
- Available both as runtime library and as I/O server configuration
  - It can serve as I/O module for high performance languages and supports the standardized MPI-IO interface
- It was developed by focusing on workstation cluster system
- Parallel I/O for High Performance Languages
  - Static fit property: adapt disk access profile of programmer or compiler
  - Dynamic fit property: adaptability of the I/O runtime subsystem to applications or environment's characteristics

29

Design Principles

- Gather all available information on the application process both during the compilation process and runtime execution
  - Provide the optimal fitting data access profile for the application and may then react to the execution behavior dynamically
  - Allow optimal performance by aiming for maximum I/O bandwidth
- ViPIOS is an I/O runtime system which provides efficient access to persistent files
  - by optimizing the data layout on the disks and allowing parallel read/write operations

30

Design Principles

- ViPIOS is supporting I/O module for high performance languages
- Design principles
  - Scalability
    - System architecture is highly distributed and decentralized
  - Efficiency
    - Suitable data organization to provide a transparent view of the stored data on disk to the outside world
    - Organize data layout on disk respective to the static application problem description and dynamic runtime requirements
  - Parallelism
    - Perform various forms of efficient and parallel data access modes

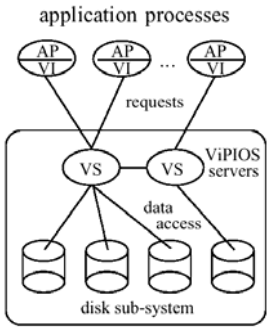
31

System Architecture

- The basic idea to solve the I/O bottleneck in ViPIOS is decoupling
  - Disk access operations are de-coupled from the application and performed by an independent I/O subsystems
  - Build upon a set of cooperating server processes
  - Server processes run independently on all or a number of dedicated processing nodes on the underlying MPP
  - One-to-many relationship between the application and the servers

32

ViPIOS System Architecture



ViPIOS

33

System Architecture

- Data Locality
  - It is design principle to achieve high data access performance
  - The data requested by an application process should be read/written from to the best-suited disk
  - Logical Data Locality
    - Choose the best suited ViPIOS server for an application process
  - Physical Data Locality
    - Determine the disk set providing the best data access

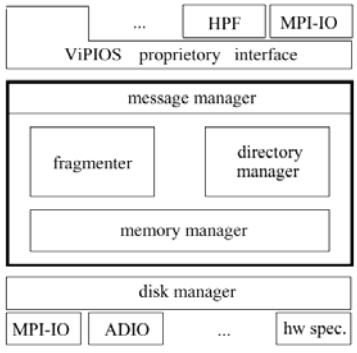
34

System Architecture

- ViPIOS Server
  - Interface layer
    - provide the connection to the outside world
  - Kernel layer
    - responsible for all server specific tasks
    - Message manager is responsible for communication externally and internally
    - Fragmenter makes decisions on the effective data layout, administration, and ViPIOS actions
    - Directory manager stores the meta information of the data
    - Memory manager is responsible for prefetching, caching and buffer management
  - Disk manager layer
    - provide the access to the available and supported disk sub-systems

35

ViPIOS Server Architecture



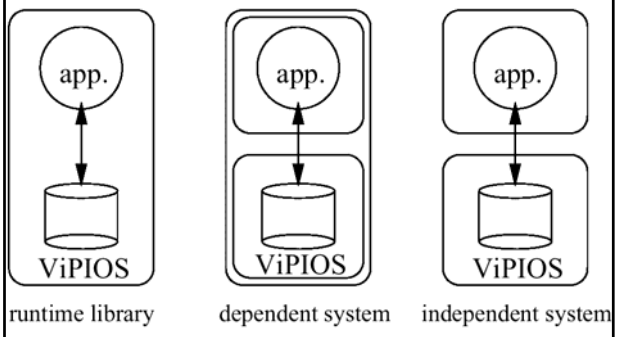
36

System Architecture

- System Modes
  - Runtime library
    - perform all disk I/O requests of the program
    - ViPIOS is not running on independent servers, but as part of the application
  - Dependent system
    - ViPIOS is running as an independent module in parallel to the application, but is started together with the application
  - Independent system
    - achieve highest possible I/O bandwidth by exploiting all available data administration possibilities
    - ViPIOS is running similar to a parallel filesystem or a database server waiting to connect via ViPIOS interface

37

ViPIOS System Modes



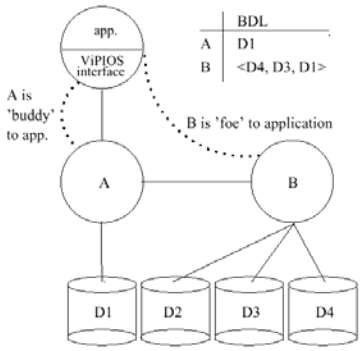
38

Data Administration

- The data administration in ViPIOS is guided by two principles
  - Two phase data administration
  - Data access modes
- Data access modes
  - Local data access (buddy access)
    - the buddy server can resolve the applications requests on its own disks
  - Remote data access (foe access)
    - the buddy server cannot resolve the request on its disks and must broadcast the request to the other ViPIOS servers to find the owner of the data

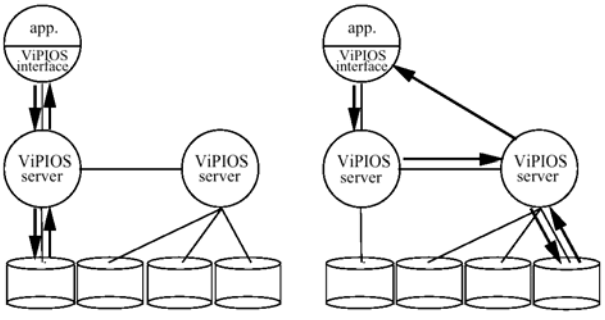
39

"Buddy" and "Foe" Servers



40

Local versus Remote Data Access



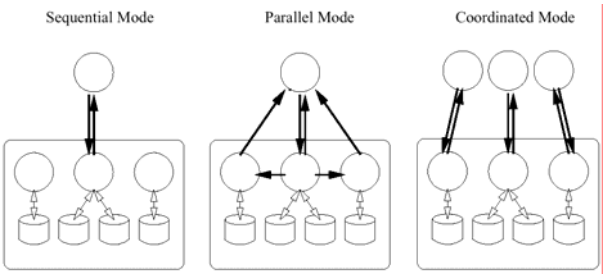
41

Data Administration

- Disk access modes
  - Sequential mode
    - allow a single application process to send a sequential read/write operation, which is processed by a single ViPIOS server in sequential manner
  - Parallel mode
    - ViPIOS processes the sequential process in parallel by splitting the operation into independent sub-operations and distributing them onto available ViPIOS server processes
  - Coordinated mode
    - a read/write operation is requested by a number of application processes collectively
    - sub-operations are processed by ViPIOS servers sequentially

42

Disk Access Modes



43