

## CS519 Operating System

### **ACTIVE DISKS:**

Programming Model, Algorithms and Evaluation

### **NASD:** (Network-Attached Secure Disk)

A Cost-Effective, High-Bandwidth Storage Architecture

## Active Disks:

Programming Model, Algorithms and Evaluation

### Basic Idea:

#### **Place computational capability on individual disks**

- Integrate processing power and memory into a disk drive
- Allow application-specific code to be downloaded and executed on the data that is being read or written
- Offload bulk of the processing to the disk-resident processors and use the host-processor primarily for coordination, scheduling, and combination of results from individual disks

## Background and Motivation

### Application end

- Needs faster rate to get data from the storage
- Needs frequent reprocessing datasets in entirety

### Technology end

- Disk transfer rate has been increasing rapidly
- Processor cannot keep all disk drives busy
- Economically feasible to place computational capability on individual disks

### Advantage of Active Disks

- Number of processors scales with the number of disks
- Processing capacity will evolve as the disk drives evolve

## Question raised by active disks

- How are they programmed?
- What is disk-resident code (i.e. a disklet) allowed to do?
- How does it communicate with the host-resident component?
- Is it feasible to utilize Active Disks for the classes of datasets that are expected to grow rapidly?
- How much benefit can be expected with current technology and in foreseeable future?

## A stream-based programming model

- Disklets take streams as inputs and generate streams as outputs
- Three types of streams:
  - *Disk-resident* streams: files or ranges in files
  - *Host-resident* streams: used by host-resident code to interact with disklet
  - *Pipe* streams: used to pipe results of one disklet into another.

## DISKLET

- Each disklet must have
  - At least one input stream and at least one output stream
  - An Initialization function which is run when the disklet is installed
  - A processing function which is run as data is read/written
- A disklet is not allowed to initiate I/O operations
  - Disklet cannot corrupt the file-system
  - The OS-layer on the disk need not to provide file-system functionality
- A disklet can skip subranges in an input stream
  - By notifying the OS-layer on the disk, the skipped subranges are not delivered to the disklet
  - Allow safely implementation of algorithms in which future I/Os depend on data from previous I/O

## DISKLET

- A disklet cannot allocate or free memory
  - A natural sandbox(input streams and long-term scratch surface) is defined for disklet
  - All memory access by a disklet must be within the sandbox
- Communication between a disklet and its environment is restricted to its input and output streams.
  - Host-resident program decides where the input stream comes from and where the output stream goes to
  - A disklet does not need handle buffering and scheduling for its communication, reducing complexity.
  - In heterogeneous environment, disklets that process data form conventional disks execute transparently on the host itself
- Installation and Invocation of disklets are separate
  - Once downloaded, disklets will be used repeatedly

## Design of *DiskOS*

(the operating system layer on disk)

### Conflicting Requirements:

- Would like DiskOS to be as thin as possible
- Want to move as much as possible of common functionality into DiskOS
- DiskOS provide three service
  - Memory management
  - Stream communication
  - Disklet scheduling
- Host-level OS
  - Support for installation of disklets
  - Management of host-resident streams

# Algorithms

## conventional-disk and active disk versions

- Relational databases
  - SQL Select
  - SQL Group-by
  - external sort
  - Datacube operation for decision support
- Image databases
  - Image convolution
- Satellite data repositories
  - Generation of earth images from raw satellite datax

## SQL SELECT

- Conventional-disk algorithm
  - Scan the entire relation
  - Apply filtering predicate independently to each tuple
  - Amenable to coarse-grain parallelization
- Active-disk algorithm
  - Apply SELECT predicate at the disk, forward only successful tuples to the host
    - Input Stream is the entire file on disk
    - Scratch-space consists of a large buffer used to collect tuples that are to be sent to the host
  - At the host, data from different disks is concatenated and transferred to the requesting client

## Generating composite satellite image

- ❖ Projection of sensor values onto a two-dimensional grid
- ❖ Composition of all values that map onto a single grid point to generate the associated pixel
- Conventional-disk algorithm
  - Process sensor values in large chunks, mapping each value to the output grid
  - Perform composition using an accumulator for every pixel
- Active-disk algorithm
  - Pre-processing and mapping at the disk
  - Perform composition locally as much as possible
  - Host maintains the accumulators for the entire image. As each sub-image is received, it is composed to the final image

## Contributing Factors

- Parallelism
  - For algorithms that perform large amounts of computation per byte and deliver their data to the host
  - E.g., Satellite data processing
- Avoiding I/O interconnect
  - For algorithms that perform little computation per byte and achieve significant reduction in the data delivered to the host
  - E.g., SQL Select
- Combination of the two factors
  - For algorithms that redistribute most of their dataset
  - E.g., sort

## Evaluation

- Active disk achieve performance improvements between 1.18 times and 3.2 times for **4-disk** configuration
- Active disk achieve performance improvement between 2.9 times and 29 times for **32-disk** configuration
- active-disk architectures retain most of their advantages even if host processor is upgraded more frequently than the disk processor