# Active Disks

*A Case for Remote Execution in Network-Attached Storage*

*Erik Riedel*

*Carnegie Mellon University*

http://www.cs.cmu.edu/~riedel

Carnegie Mellon

**Parallel Data Laboratory**

# Introduction

- Trends
  - » processing power at storage is increasing
  - » bottlenecks are in other parts of the system
- Opportunity
  - » allow application-specific code to execute inside storage devices
  - » use shipped functions at storage to offload network and client/server work

# Outline

- Trends
- Opportunity
- Potential applications
- Experiment
- Mechanisms
- Conclusions & future work

# Trends

- Increased processing power on drives
  - » 100 MHz RISC core coming soon
  - » not involved in fastpath processing
    - – lots of idle cycles
    - – needs "value added" work to do
- System bottlenecks shifting
  - » drive throughput is not the major problem
    - – network utilization
    - – client/server processing

# Trends (2)

- Majority of aggregate CPU (and soon memory?) in a system is at the disks

- Microsoft TerraServer
  - » 4-CPU AlphaServer 4100
    - – (4 x 400 = 1,600 MIPS)
    - – 2,048 MB RAM
  - » 320 disks (1.3 TB)
    - – (320 x 25 = 8,000 MIPS)
    - – (320 x 1 = 320 MB)

- Compaq ProLiant TPC-C
  - » Four 200 MHz Pentium Pros
    - – (4 x 200 = 800 MIPS)
    - – 4,096 MB RAM
  - » 113 disks (708 GB)
    - – (113 x 25 = 2,825 MIPS)
    - – (113 x 1 = 113 MB)

» largest part of system cost is the storage

# Opportunity

- Candidate applications
  - » can leverage the available parallelism
    - – highly concurrent workloads
    - – lots of drives compensate for lower relative MIPS
  - » are localized to small amounts of data
    - – process as data “streams past”
  - » have small code/cycle footprint per byte
  - » can use scheduling, management primitives
    - – enable a new range of storage functions

# Opportunity (2)

- Classes of applications
  - » filtering - search, association matching, sort
  - » batching - collective I/O
  - » real-time - video server
  - » storage management - backup, layout
  - » specialized support - locks, transactions

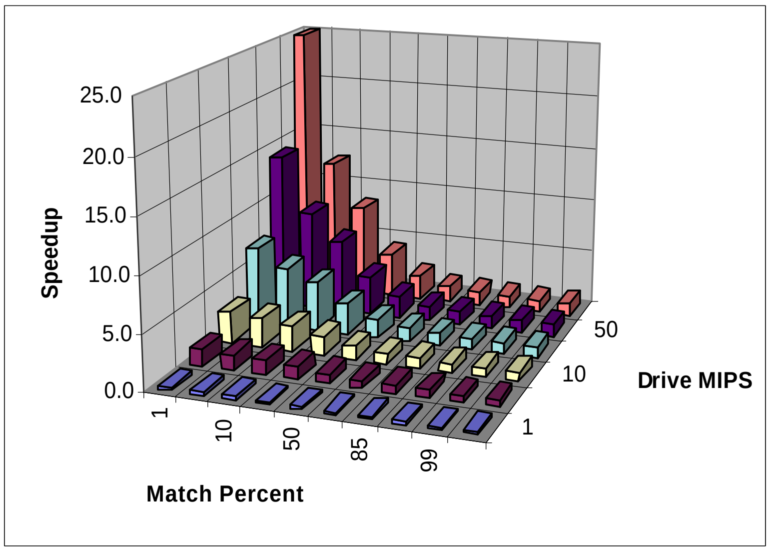
} *scheduling*

# Applications - TIP Suite

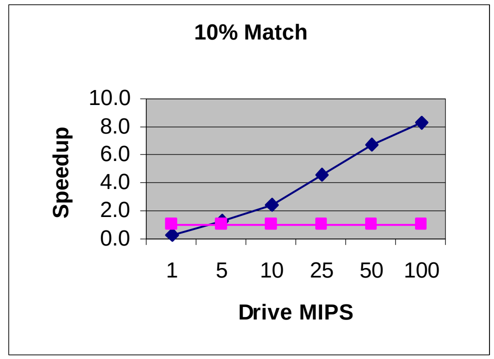
- Reduce data transfer w/ “low” cost
  - » *agrep* - significant filtering
  - » *xDataSlice* - some filtering
  - » *gnuld* - expensive computation
  - » *Sphinx* - cpu intensive
  - » *Postgres*
    - – indexed join - poor locality w/o hints
    - – unindexed select - good filtering

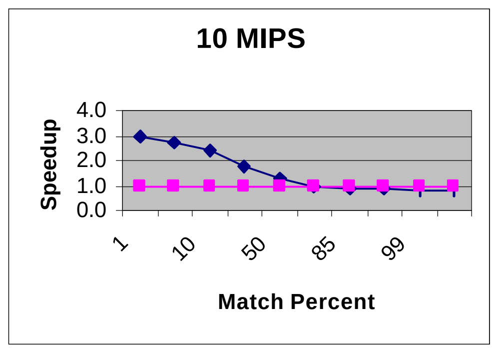
# Applications - Database Select

» varying match percentage and drive MIPS[1]




[1]Underlying numbers from [Franklin, Jonsson, Kossman] in SIGMOD96



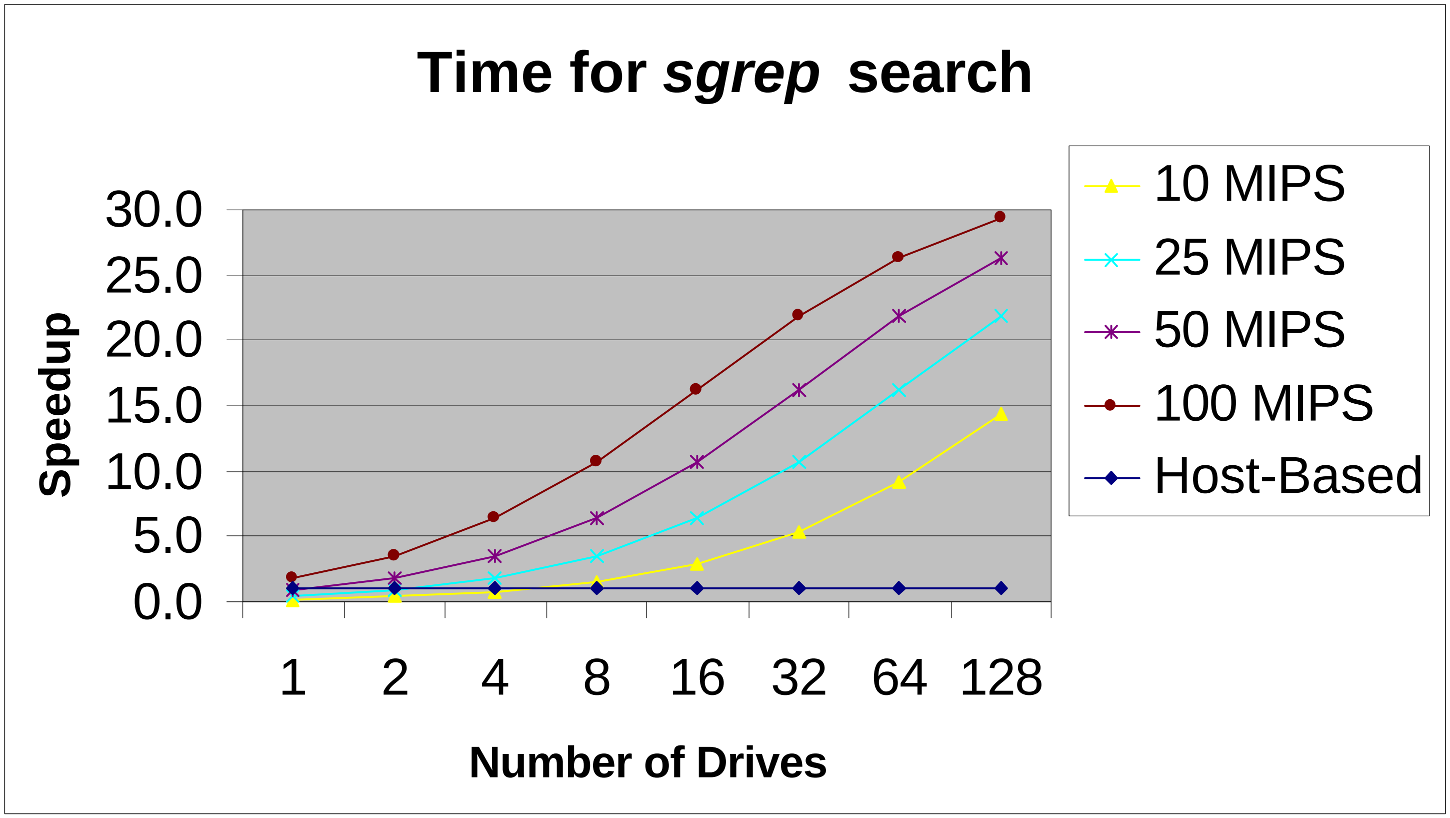




– considers only CPU cycles
  - assumes excess drive bandwidth
  - network link is the bottleneck
– speedup vs. a 200 MIPS host

» when match% is low gains are possible even with only 10 or 25 MIPS drives

# Applications - *sgrep* Search

» varying drive MIPS and parallelism




– speedup vs. 200 MIPS host

# Experiment - *SampleSort*

- Two stage parallel sort
  - » sample data
  - » create distribution histogram
  - » distribute data to clients based on histogram
  - » sort locally at clients
  - » write back in sorted order
- Observation
  - » filter operation on key ranges

# Experiment - *NasdSort*

- Implementation on NASD prototype
  - » two simple functions “shipped” to drive
    - – *sample()*
      - read() request that returns only a subset of the data
    - – *scan()*
      - read() request that returns data for a specific key range
      - buffers data from other ranges for later requests
  - » single master collects samples
  - » synchronization handled at the drives

# Experiment - *NasdSort* (2)

- Future extensions
  - » larger data sets
    - – add a merge phase at the end
  - » perform entire sort at drives
    - – more complex than *scan()* and *sample()*
    - – requires more cycles
    - – requires additional memory
  - » examine other sorting algorithms
    - – different scheduling characteristics

# Mechanisms

- Execution environment
  - » protect the drive and data
    - – against corruption/“leaks”
- Programming environment
  - » how to specify remote code
    - – how to “split” applications in the brave new world
- Resource management
  - » competition within the drive
    - – sector bandwidth, cache space, processor cycles

# Mechanisms (2)

- Execution environment
  - » compilation vs. translation vs. interpretation

| Technology | Per Program | | Per Invocation | | Per Statement | |
|---|---|---|---|---|---|---|
| | Cost | Where | Cost | Where | Cost | Where |
| **Compilation** | high | drive | none | | none | |
| **Pre-Compilation** | high | producer | none | | none | |
| **Sandboxing** | none | | high | drive | low | drive |
| **Interpreter** | none | | medium | drive | high | drive |
| **PCC** | high | producer | low | drive | none | |

# Mechanisms (3)

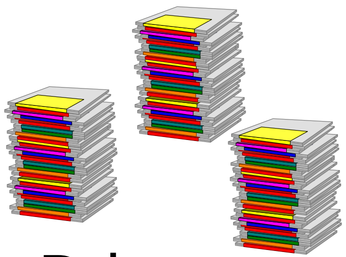
- Internal drive interface

| Functionality | Filter | Video Stream | Batching | Management | Transactions |
|---|---|---|---|---|---|
| basic filesystem API | X | X | X | X | X |
| stdin/stdout to requestor | X | | | | X |
| asynchronous "callbacks" | | X | X | | |
| long(ish)-term state | ? | X | X | ? | |
| time/deadlines | | X | X | | |
| real-time scheduler | | X | | | |
| admission control | | ? | | | |
| drive internals - query cache | | | X | X | |
| internals - query layout | | | X | X | |
| internals - control cache | | | | X | |
| internals - control layout | | | | X | |
| internals - control ordering | | ? | X | X | ? |
| internals – "eavesdrop" requests | | | | X | |
| initiate commands to 3rd parties | | | ? | ? | ? |
| object locks/atomicity | | | | | X |

# Resource Management

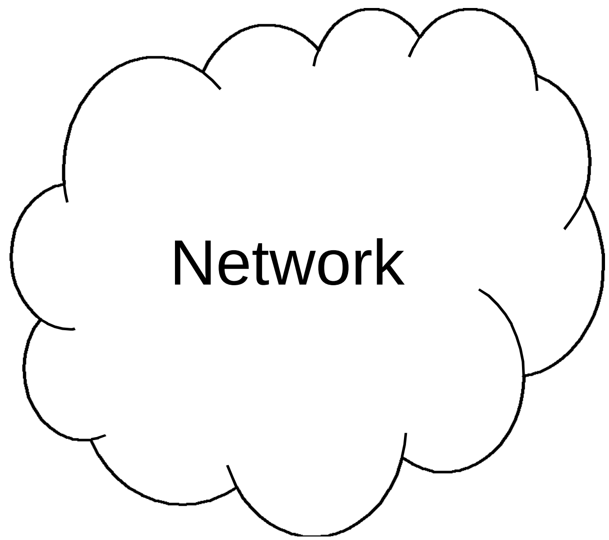
- How to “control” the impact at drive
  - » limit functions to the cost of a “normal” op
    - – allow 2-3x the resources of a *read()* operation
  - » allow functions only during “idle” periods
    - – problematic in the presence of prefetching e.g.
  - » allocate a specific amount of resources to RE
    - – allocate that among all active functions
  - » TIP-like model cost/benefit
    - – minimize total application wait

# Optimal Partitioning

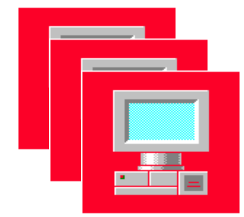


## Drives

- sector bandwidth
- cache memory
- processor cycles
- program memory




- bandwidth
- number of messages
- congestion
- connection setup/teardown
- data integrity/protection



## Clients/"Servers"

- processor cycles
- cache memory
- deadlines
- request "state"

# Conclusions

- Significant “free” processing capability available on storage devices
- Potential for improving performance across a range of application classes
- Opportunity for value-add directly at storage devices

# Future Work

- Resource management
  - » admission control for shipped functions
- Trusted environment
  - » pre-compilation for safety
- Storage management applications
- Additional domains
  - » data warehousing
  - » web servers

# Related Work

- Active technologies
  - » Active Networks (MIT), Liquid software (Arizona), Postscript (Adobe)
- Database technologies
  - » Hybrid-shipping (Maryland), nowSort (Berkeley), Parallel database systems, Database machines, Channel programs

# Related Work (2)

- Extensible operating systems
  - » SPIN (Washington), exokernel (MIT), VINO (Harvard), Scout (Arizona), Synthetix (OGI)
- Language technologies
  - » OmniWare (Berkeley/CMU), Toba (Arizona), Javelin (Santa Barbara), Inferno (Bell Labs), Proof-Carrying Code (CMU)
- Object Technologies
  - » CORBA, DataBlades (Sybase), DCOM