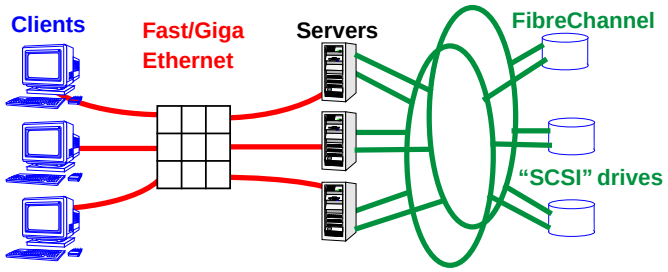
Active Storage Nets

David Nagle
Computer Engineering and Computer Science
Carnegie Mellon University

Network-attached storage is an excellent application for Active Storage Nets



Network-Attached Secure Disk (NASD)

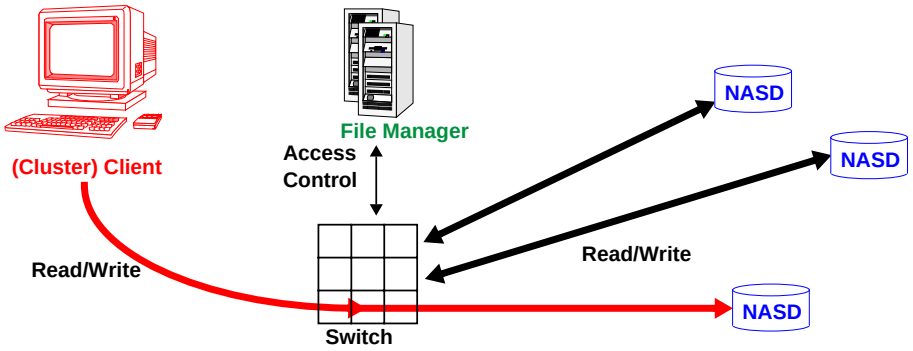
Enable direct transfer between client & storage device

Object-oriented interface (SCSI 4) w/persistent objects

Objects supported by in-drive file system

File manager provides policy (e.g., names, ACLs, consistency)

Device understands cryptographic capabilities



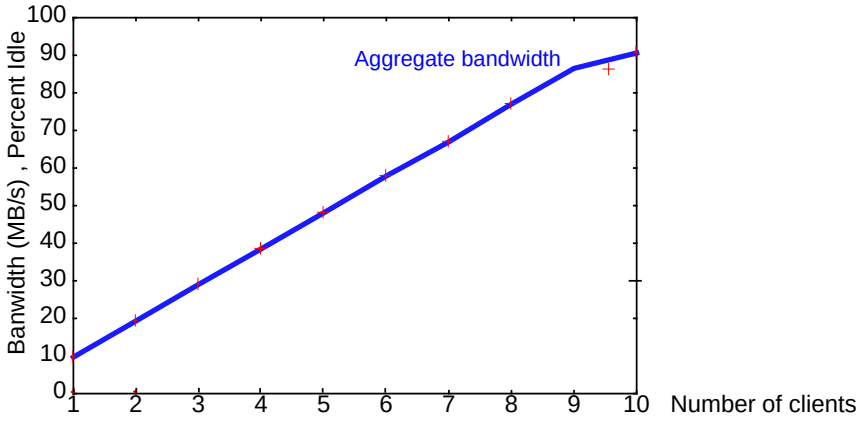
Current NASD Prototype

Current prototype supports NFS, AFS and Striped NFS

Object Interface (SCSI-4) moving toward standard

Storage management middleware in development

NASD/NSIC Working Group: CMU, HP, Quantum, IBM, StorageTek, & Seagate



Beyond Bandwidth: all the other goals

Storage self-management - 3-7X capital cost/year

Object-oriented disks - performance optimizations

Quality of storage - object attributes and scheduling

Securable devices - reduce trust scope

Scalable cluster servers - multi-device serializability

Avoid OS stranglehold - user-level file/storage systems

Enable nimble vendors - extensions unknown to FS

Active disks - application exploitation of device cycles

Potential Issues with NASD

Network protocols from SAN, LAN and WAN

- SAN traffic fast ... WAN traffic slower ... What about LAN traffic
- Drive supports multiple protocols or uses protocol converter (e.g., server)

Storage Function (RAID, Striping)

- Central coordination point has been the server or RAID controller

Security

- Costly ... why repeat at multiple levels in the system
- What happens to security when NASD objects are cached

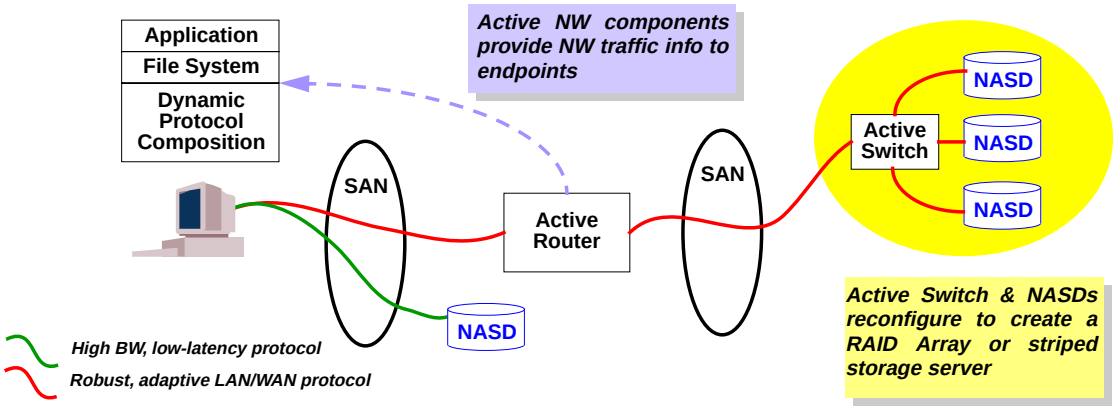
On-drive resources

- Limited on-drive resources: limited RAM ... no extra SIMM slots on a drive
- Future network-attached storage legacy systems not as powerful

Active Nets + NASD

What can Active Nets do for Network-attached Storage?

- Dynamically composed and adaptive storage protocols
- Network components aware of NASD objects
  - Caching, reliability, security, & prefetching via NASD objects
  - Quality of Service meets Quality of Storage
- Network-embedded storage functionality



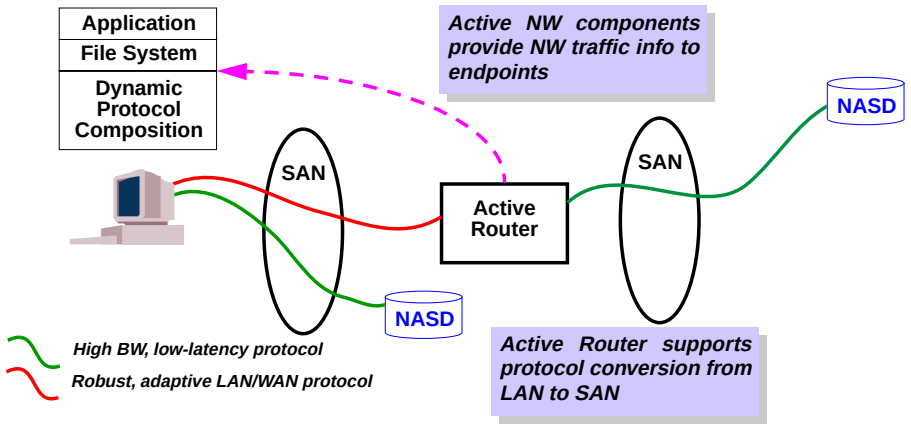
Protocol Support

SANs enable lightweight protocol ... WAN needs TCP/IP

- LAN traffic protocol based on location, physical NW & application

NASD support for TCP/IP can be difficult

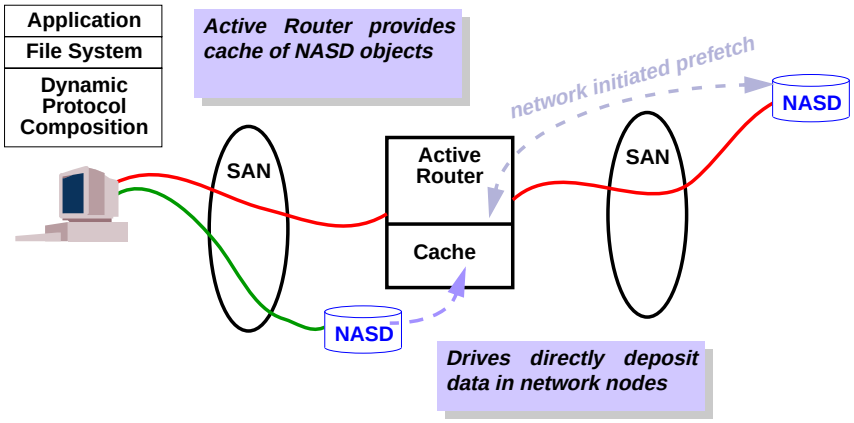
- CTCP-like protocol pushes more function to the client [Johnson98]



NASD Object Aware Networks

Drive object model provides standard access to data

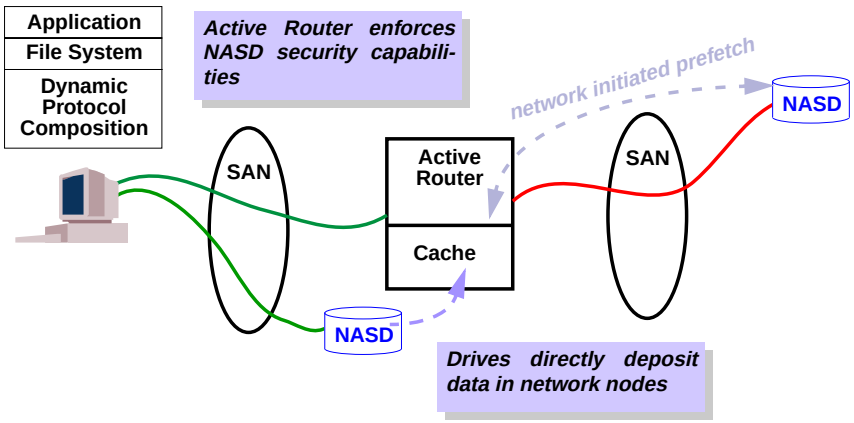
- Network nodes cache/manage complete objects ... or groups of objects
- Network nodes can initiate cache and prefetching algorithms
- Drives (and clients) can expand RAM by using network nodes



Security

NASD-aware networks can support NASD security

- Net caches can enforce NASD capabilities
- Allows storage to transparently extend security model into network



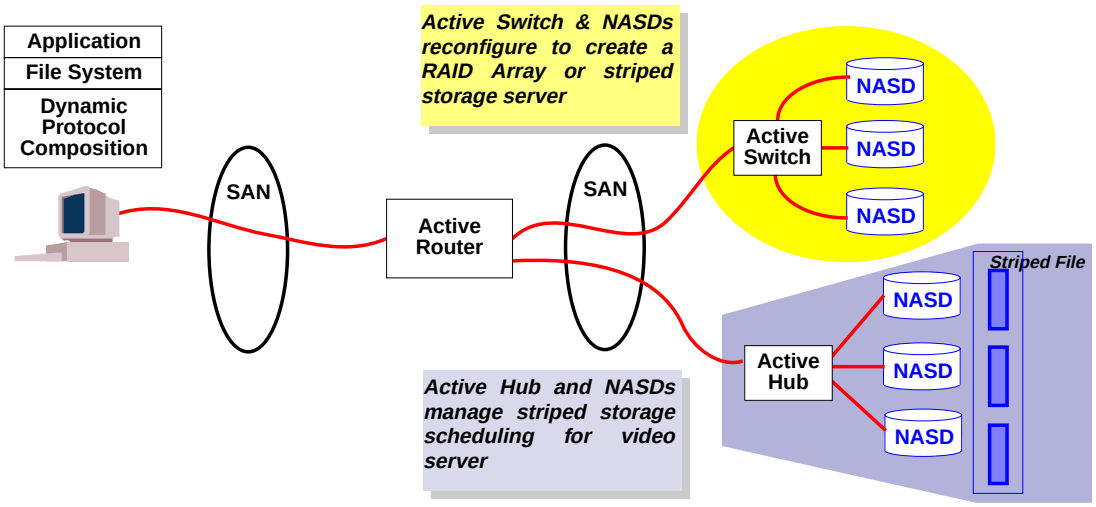
Net-based Storage Functionality

RAID and striping are object-based not drive-based

- Client middleware provides storage function

Active Nets can accelerate storage function

- Network coordinates amoung independant devices to provide QOS

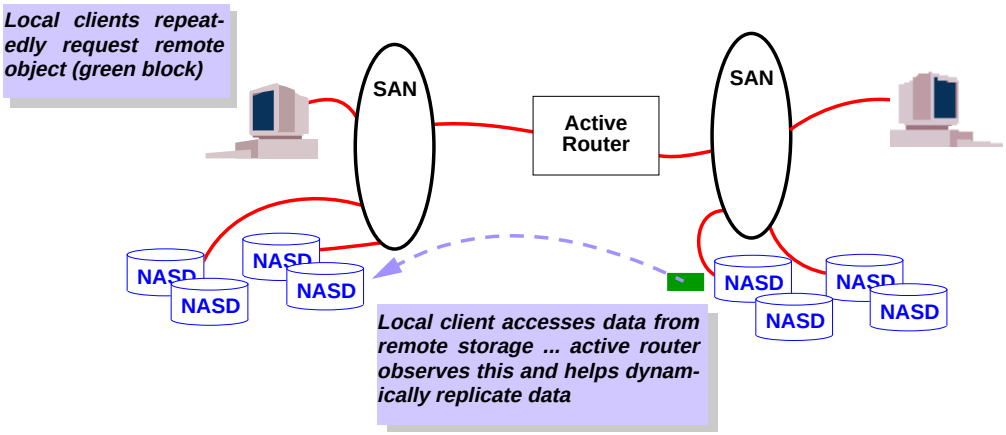


Dynamic Replica Management

Create dynamic replicas for minimizing WAN traffic

Dynamically copy or move data in LAN and SAN based on application behavior

- Network and storage coordinate to deliver QOS



Summary

Infrastructural computing via Active Nets & NASD

- Active networks enable storage access from diverse locations
- NASD-aware networks extend storage capabilities/functionality
- Active nets programmability enables flexable storage management

Current work

- Integrate Active Nets and NASD testbeds
- Designing adaptive protocols and storage API
- Storage function in the net

Tech Transfer

NASD/NSIC working group: IBM, Seagate, Quantum, StorageTek, HP

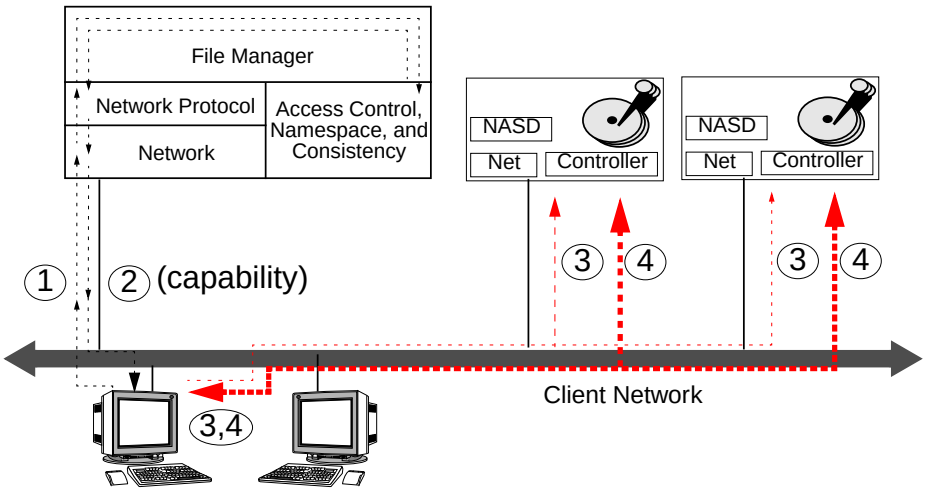
Parallel Data Consortium: HP, Seagate, Quantum, StorageTek, Intel, Compaq, 3Com, Symbios, Clariion, & WindRiver

CMU NASD is an Object Oriented Disk

Avoid file manager unless policy decision needed

- access control once (1,2) for all accesses (3,4) to drive object
- spread access computation over all drives under manager

Scalable BW, off-load manager, "file" knowledge



CMU's Functional Definition of NASD

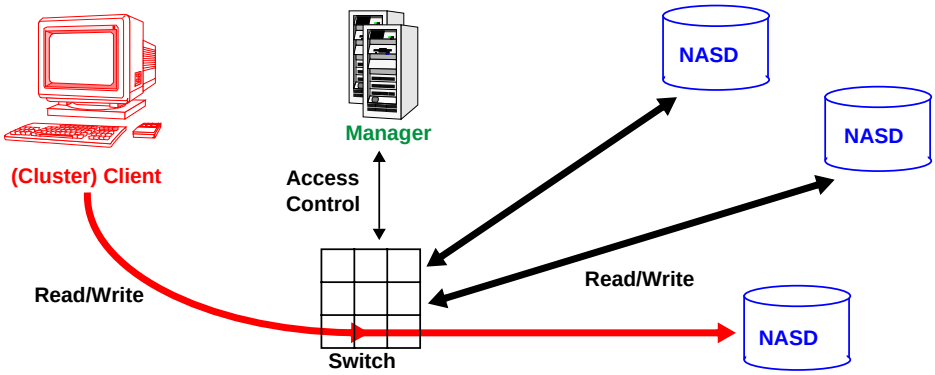
Direct client/drive transfer in networked environment

Asynchronous filesystem oversight of rights, semantics

Cryptographic capabilities ensure command integrity

Self-management by more abstraction, independence

Extensible features for application, not just client OS



NFS Motivation for NASD Design

NASD common code path: read bytes from file FOO

send <read, bytes, capability(FOO=objX)> to NASD
NASD: lookup objX, verify capability, fetch bytes from objX
reply <read-reply, data> directly to client

NFS Operation	Count in top 2% by work (thousd)	SAD		NetSCSI		NASD	
		Cycles (billions)	%of SAD	Cycles (billions)	%of SAD	Cycles (billions)	%of SAD
Attr Read	792.7	26.4	11.8%	26.4	11.8%	0.0	0.0%
Attr Write	10.0	0.6	0.3%	0.6	0.3%	0.6	0.3%
Block Read	803.2	70.4	31.6%	26.8	12.0%	0.0	0.0%
Block Write	228.4	43.2	19.4%	7.6	3.4%	0.0	0.0%
Dir Read	1577.2	79.1	35.5%	79.1	35.5%	0.0	0.0%
Dir RW	28.7	2.3	1.0%	2.3	1.0%	2.3	1.0%
Delete Write	7.0	0.9	0.4%	0.9	0.4%	0.9	0.4%
Open	95.2	0.0	0.0%	0.0	0.0%	12.2	5.5%
Total	**3542.4**	**223.1**	**100.0%**	**143.9**	**64.5%**	**16.1**	**7.2%**

Berkeley NFS traces [Dahlin94] (230 clients, 6.6M reqs)

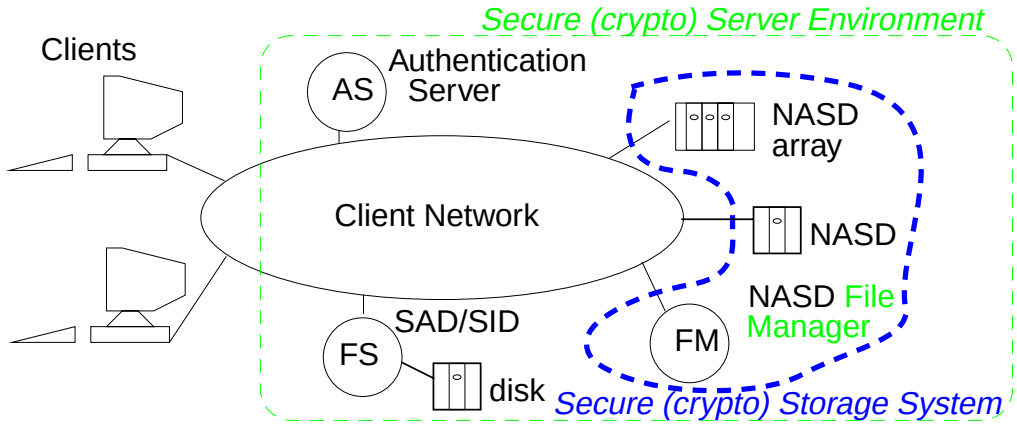
Security implications of NASD

Not tied to any specific higher-level security system

ie., not Kerberos, authenticated RPC, x.509

Authenticates command to be file manager approved

rests on secrecy of file manager key (hierarchy) only
anticipates hardware support in drive ASIC

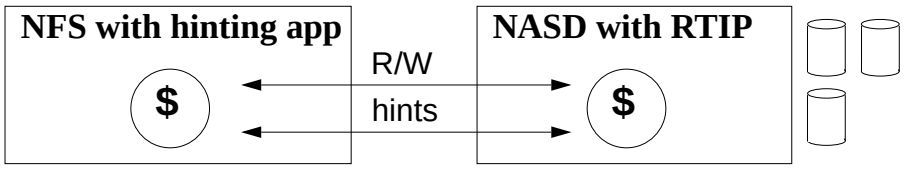


NASD object i-face: Where's metadata?

Not at client: don't rest integrity on trusted client

Data Layout in storage device?

- avoid distributing per-drive block lists to drive
- enables on-drive, drive-subsystem optimization
- ie. interposed/stackable NASD - striping, RAID
- ie. remote Transparent Informed Prefetching



- **3D visualization rendering random 25 planes**
- **120 secs w/o hints and 68 seconds with hints**

NASD Interface Design: Storage Objects

Layout is best (actually) done below SCSI-4

- real-time support possible; accurate geometry
- simplifies, strengthens transparent performance optimization

Drive serves storage objects on behalf of manager

- Objects have attributes
  - Inodes: (name), size, protection, type, timestamps, layout
- Object layout guidance from higher level
  - sequential within object requests contiguous (can preallocate)
  - related objects can be clustered with "nearby to" attribute
- Attributes are extensions of object name
  - give higher level (filesystem) uninterpreted attribute
  - big enough to contain another object name (soft link)

Supporting File Managers: Soft Partitions, Well-known Objects

Split capacity among different managers: partitions

- managers can use attributes differently; no need to integrate
- boundary should be soft: resize partition should be fast
- partition key hierarchy: (partition key, working keys)

Well-known objects replace SCSI mode sense/select

- published format and interpretation
- Per-drive and per-partition separated
- eg. Partition table of contents (mini-disk directory)
- eg. Assist simple boot code with easily found "first object"

Adapting Filesystems to NASD

Reorganize decomposition of function (aka port)

Primitives become drive responsibility

- data transfer, synchronous/automatic metadata updates

Policy remains manager responsibility

- namespace definition/navigation
- access control policy
- client cache management
- multi-access atomicity

Managers retain control through capabilities

- exploiting attributes for naming and revocation
- restricting client operations to protect "set attribute"

Mapping Filesystem to NASD Objects

Objects: attributes, access control, clustering

Simple model

- each file and directory bound to separate NASD object
- file attributes inherit object attributes (times, logical size)

Multiple objects per file?

- internal structure: database pages, mpeg group-of-pictures
- NASD striping, redundancy

Multiple files/directories per object?

- probable contiguity, prefetching; shared metadata overhead
- capabilities can be restricted to object region

NFS, AFS simple model; Striped NFS multiple per file

Prototyping NASD: NFS & AFS on NASD

File -> NASD object; Directory -> NASD object

NASD object: private metadata, exposed attributes

- allocation: length, blocks used; times: create, data modify
- FS specific: NFS: owner, group, mode
- FS specific: AFS: above and modify time

Operation disposition

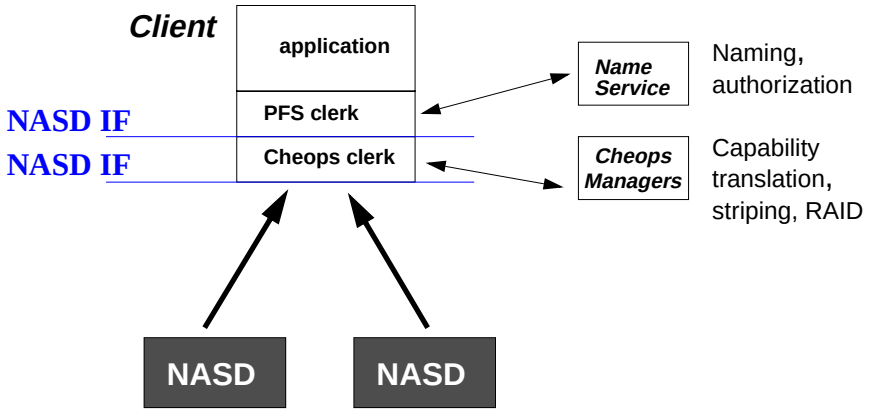
- NFS: to drive: get attribute, read, write
- AFS: to drive: FetchStatus, BulkStatus, FetchData (w/cap), StoreData (w/cap)
- AFS: Read w/o cap: GetCap (callback, attributes), (GetAttr from drive), FetchData
- AFS: Write w/o cap: GetWCap, StoreData, ReturnCap
- AFS cache coherence protocol independent of NASD
- AFS quota enforced by capability escrow (using write range)

Cheops: Striping storage middleware

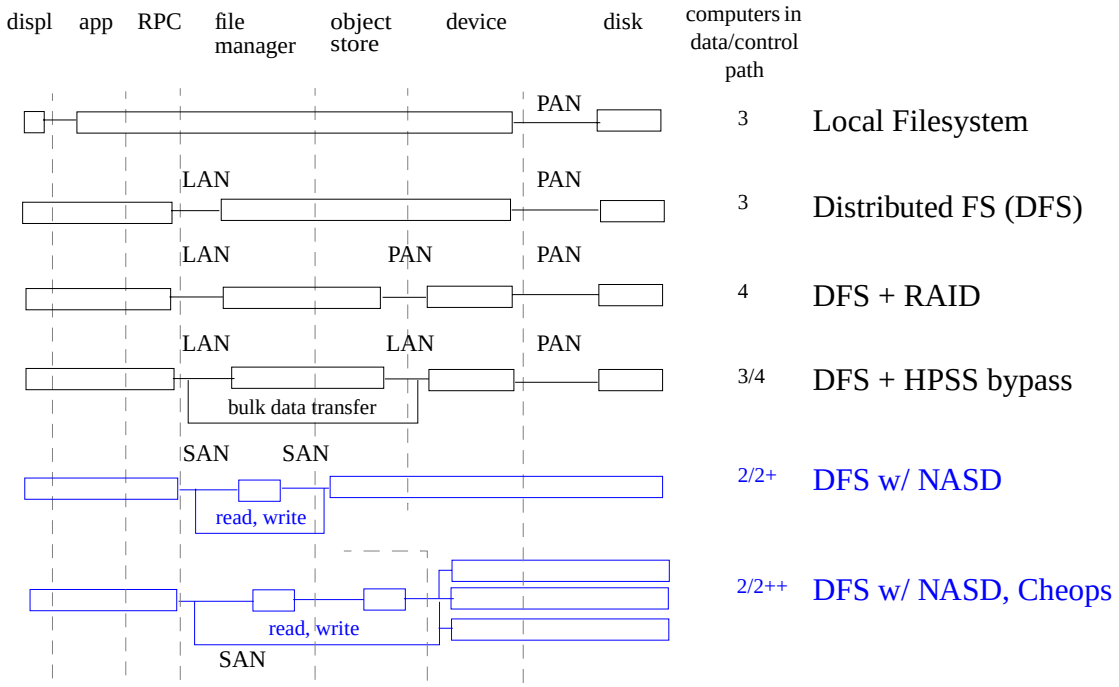
Asynchronous storage management oversight

Client asking for service pays for it (synchronizer)

- striping, RAID, incremental growth, consistent caches
- fast (optimistic) synchronization support in NASDs
- anticipates user-level network access (Intel VIA)



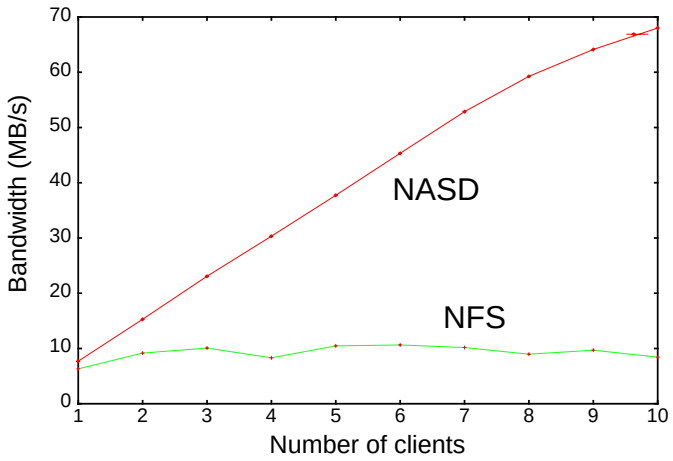
Storage Interface Evolution & Taxonomy



Scalable Bandwidth for Applications

Client library implementation enables aggressiveness

- parallel file system accommodates parallel nature of scaling
- NASD middleware fetches large blocks in parallel



Parallel Data Mining

- 13 NASDs (133Mhz)
- 1-10 clients (233Mhz)
- MPI + SIO LLAPI
- Cheops + NASD
- 7.4 MB/s per client
- until drive limits
- switched ATM LAN

Recap: SCSI-4 based on NASD

Direct transfer for wire-once, scalable bandwidth

- NASD for object bandwidth and server off-loading
- ie. Data mine at 65 MB/s in distributed filesystem

NASD filesystems serve policy (async oversight)

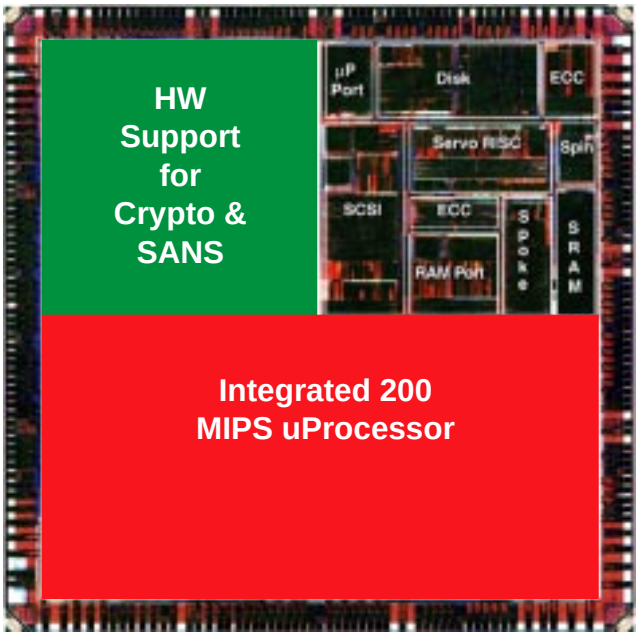
- namespace, access control, consistency, atomicity
- capabilities encode policy, metadata; crypto integrity
- NASD is attribute-based, variable length object

Storage management middleware

- clients pay for requested synchronizing semantics
- striping, RAID, incremental capacity, migration
- optimistic synchronization using fs-specific attributes

Are Device Cycles Really Available?

Quantum Trident drive

- Control: M68020
- Datapath ASIC
- 4 indep clock domains, each 40 MHz
  SCSI processor
  disk R/W channel
  uP control port
  DRAM port
- ~110 Kgates + 22Kb
- .35 micron next gen. enables integration of control uP onto ASIC
- Siemans Tri-Core Architecture

Current .68 micron chip is 74 sq. mm