Parallel Data Systems

Storage & Computer Systems Integration in DSSC

Efficient integration of storage systems into high-performance computing environments

- meet storage needs of tomorrow's computer systems
- develop strategies for systems to exploit new technology

Drawing on research from four relevant areas:

- redundant disk arrays (G. Gibson)
- fault tolerant computing (D. Siewiorek)
- distributed file systems (M. Satyanarayanan)
- parallel supercomputing, fast networks (HT Kung, P. Steenkiste)

Laboratory Research Foci

1) Reliable/Available Storage with High Throughput

- highly efficient use of storage devices
- tunable reliability vs availability vs performance vs cost
- effective integration with delivery systems
- wide exploitation of parallelism

2) Beating the Access Gap: Low Latency Data Reads

- reduction of time applications block awaiting I/O
- hiding storage device access latency
- extracting parallelism from serial access patterns
- converting high throughput systems to low latency systems

Reliable/Available Storage with High Throughput

Explosive growth in computer processing speeds

- 40-100% per year in microprocessors/VLSI
- up to 150% per year from parallel or multi-processing
- storage system throughput must also grow explosively!

Key Technology: DISK ARRAYS

Scale throughput with device parallelism

- explore massive disk arrays
- develop algorithms for more effective disk arrays
- explore integration with modern operating systems, networks, and applications

Throughput: Immediate Plans

Failure Recovery Systems: High Availability

- **explore tradeoffs: cost, availability, reliability, and performance**
- **develop algorithms for failure recovery tunable to tradeoffs**

Parity Logging: High Throughput OLTP

- **minimize performance penalty of maintaining redundancy**
- **explore implications for availability, reliability, massive arrays**

Nectar-based Disk Array: Experimental Testbed

- **high performance computing environment: iWarp, Cray, Alpha**
- **implement and evaluate recovery algorithms, parity logging, informed prefetching, and alternative disk array design issues**

Storage Server Requirements: Research Platform

Nectar Applications

- stimulate new users - grand challenges applications
- measure and model high-performance server workload

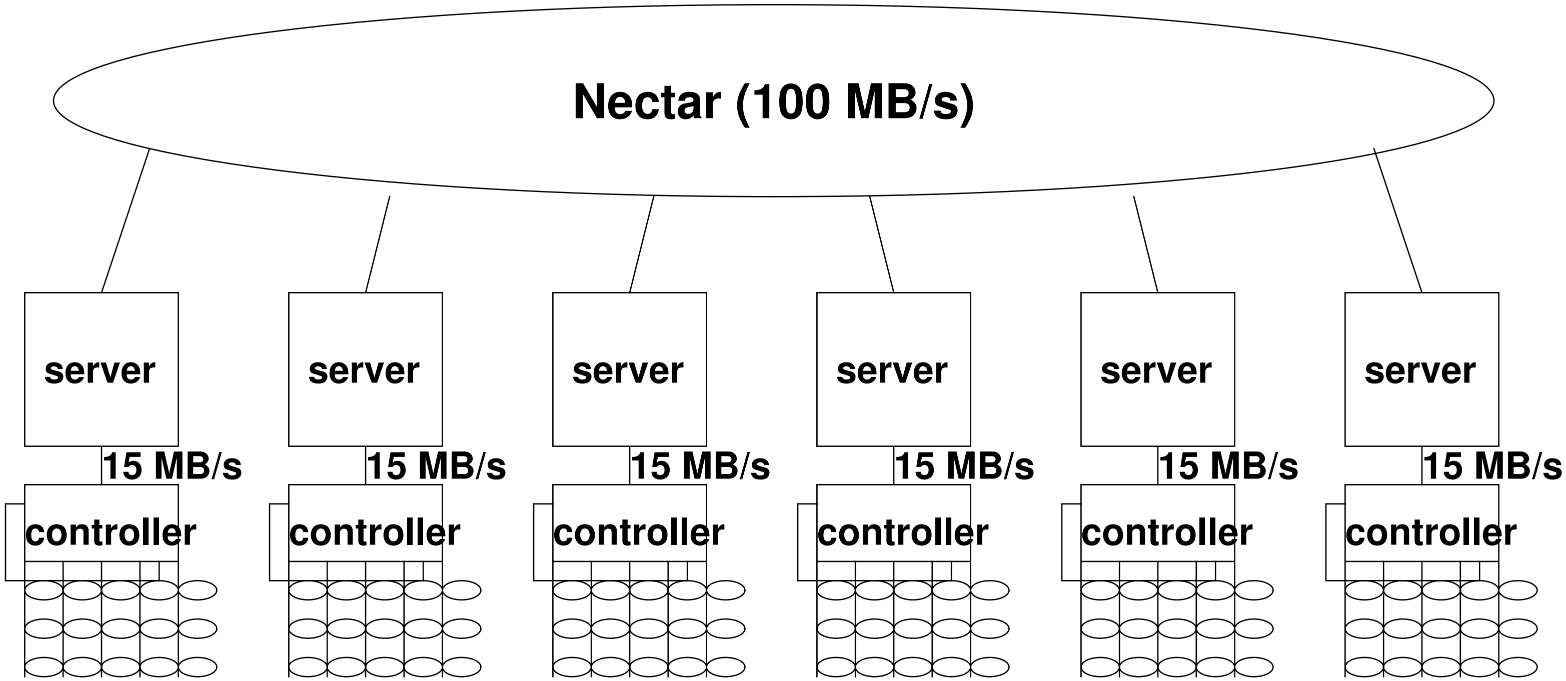
Network Storage Server Architecture Issues

- maximize efficiency and scalability of most prevalent use of network

Laboratory Proving Ground

- evaluate recovery algorithms, parity declustering, parity logging
- demonstrate and facilitate informed prefetching

Nectar Storage Server Architecture



6 X 10-30 MB/sec = 60-180 MB/sec

6 X 10-50 drives = 90-300 drives

6 X 10-35 GB = 60-200 GB

Throughput: Long Range Plans

Massive Disk Arrays built of Small Disks

- **develop models of small disk technology and interfaces**
- **develop disk array architectures for exploiting small disks**

Distributed Disk Arrays Scale through Network

- **partition storage system over many links/nodes of network**
- **develop systems for controlling distributed, parallel storage**

- **SCALABLE, MASSIVE DISK ARRAYS**

Beating the Access Gap: Low Latency Data Reads

Application not scaling with processing speed

- fixed access sizes - diminishing returns from increasing parallelism
- fixed access patterns - diminishing returns from larger file caches

Key Technology: PREFETCHING

Extract future accesses to exploit maximum storage throughput

- develop file system that prefetches according to hints
- explore benefits to important applications
- develop automatic extraction of hints
- explore integration with modern operating systems, disk arrays

Low Latency: Immediate Plans

UNIX/OSF-1 Experimental Implementation

- **develop file system implementation exploiting parallel disks**
- **evaluate system on Nectar-based disk array testbed**

Instrumentation of Important Applications

- **develop paradigm for generating/interpreting hints**

  Disclosure, advice not!

- **data search, compilation, visualization, scientific simulation**

Automatic Extraction of Hints

- **optimizing compiler techniques for describing I/O access patterns**
- **evaluate across a broad, naive class of applications**

Low Latency: Long Range Plans

Controller design for aggressive prefetching

- accommodate and exploit huge queues of accesses
- develop appropriate scheduling and prioritizing
- reduce OS interaction for non-demand access: multi-thread?

Main memory resource management policies

- main memory is throttle on aggressive prefetching
- explore balance between memory needs of virtual memory, recently used data blocks, prefetched data blocks

Advanced hint collection strategies

- robust, graceful degradation given poor or incorrect hints
- history-based hint generation