Latency Management and Quality of Service in Storage Systems

Rodney Van Meter, USC/ISI
rdv@isi.edu

Workshop on R&D Opportunities in Federal Information Services

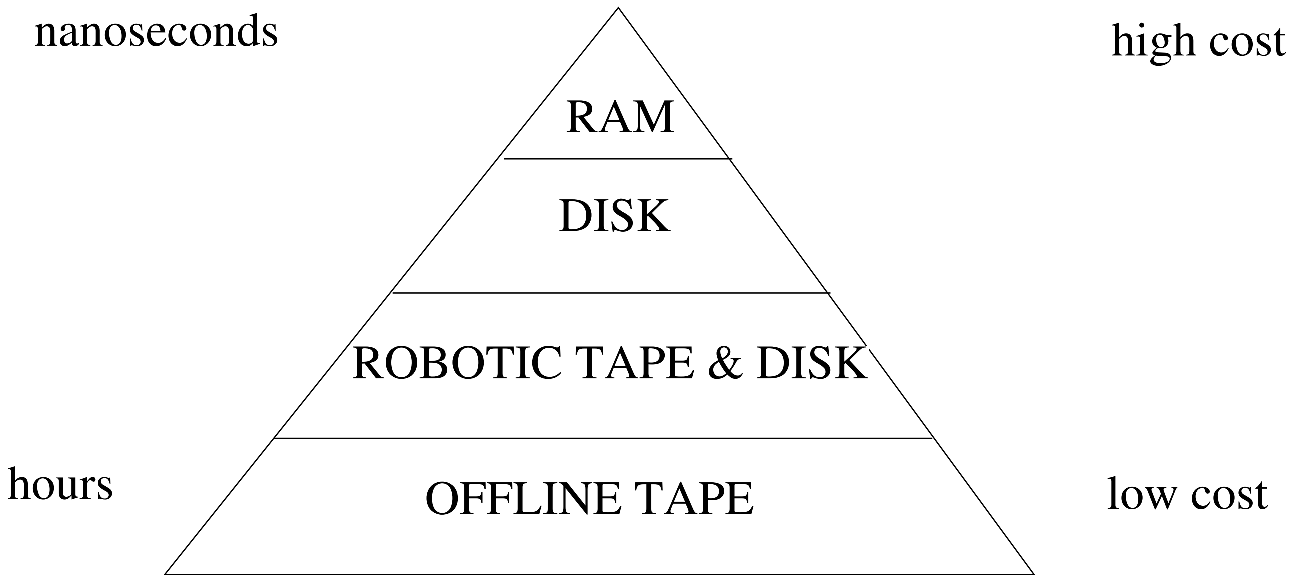
May 14-15, 1997

Effectively utilizing petabyte-class archives will require applications to cooperate with systems.

Outline

- Motivation
  latency - state problem
  trends
- SLEDs
- Impact
- Conclusions

Hierarchical Storage Management



transparent interface simplifies use

but:

programming transparency != performance transparency

Dynamic Storage State

- local file system buffer cache
- server cache
- disk head position
- tape head position
- removable media mount state
- autochanger position and wait queue
- network latency and bandwidth

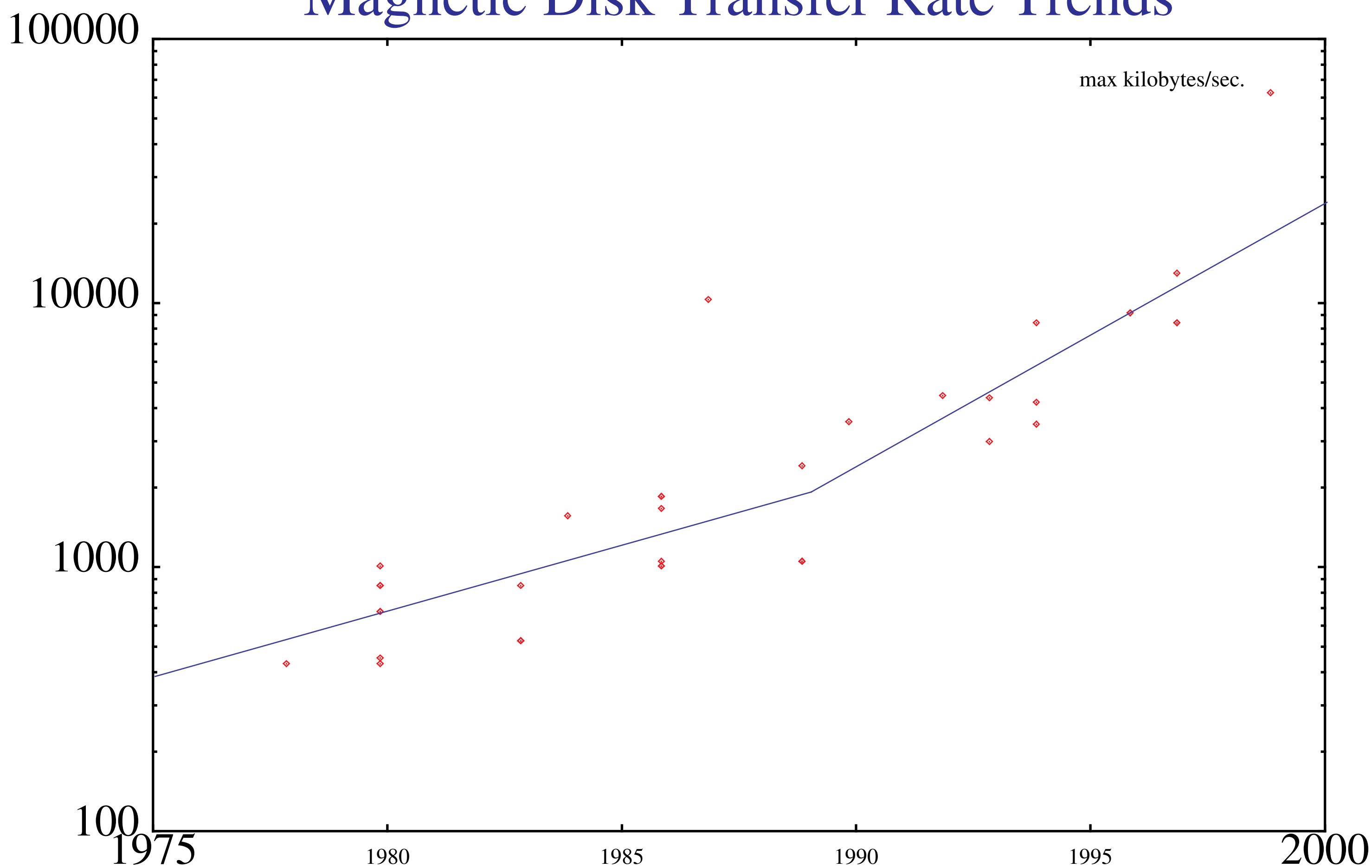
12 orders of magnitude difference, all hidden from the application!

Storage Subsystems Trends (anecdotes)

while CPUs getting faster ~55%/year since 1986...

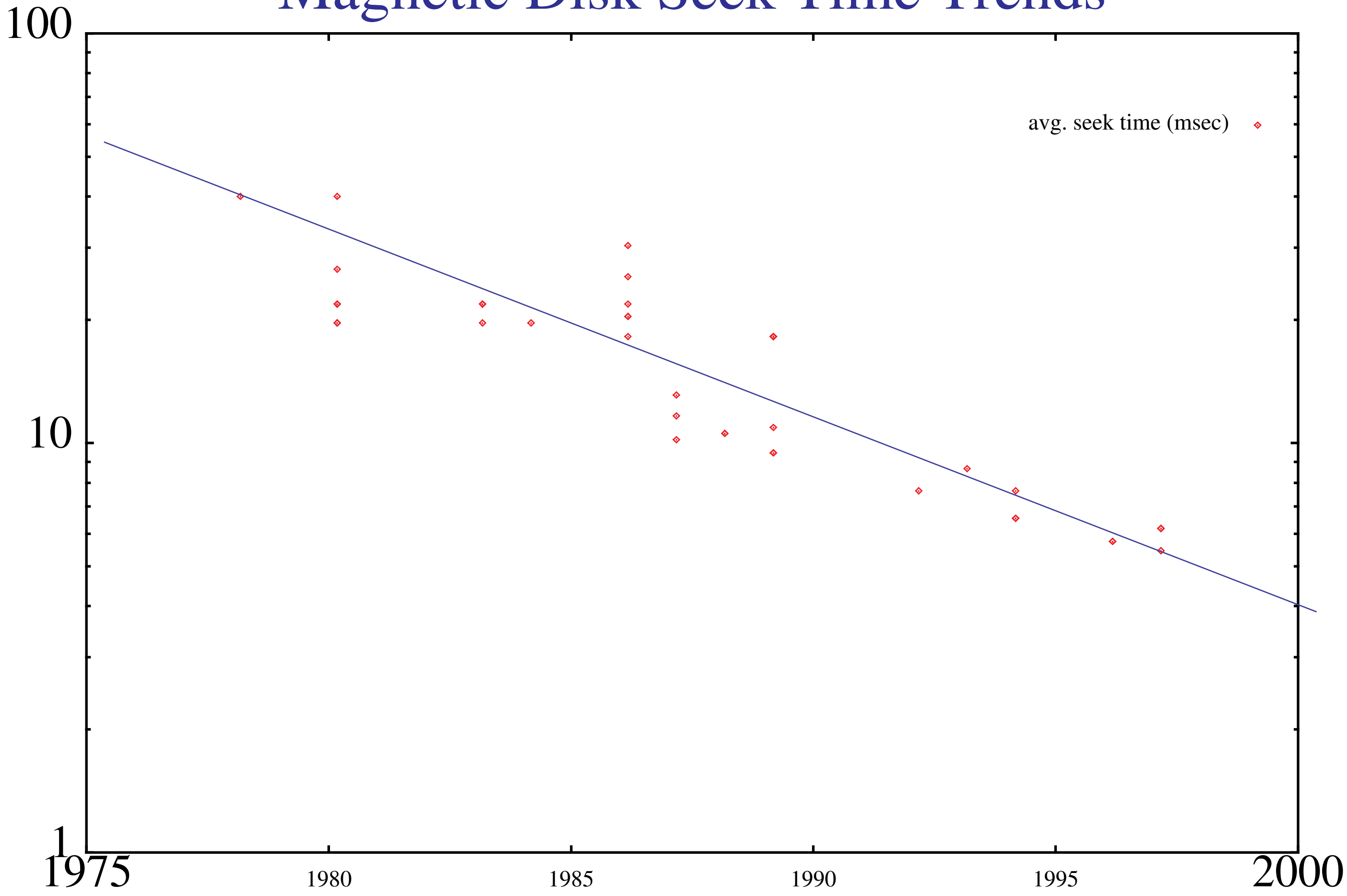
- disk transfer rates improving ~25%/year since 1989
- disk seek times improving ~10%/year
- rotational latency improving only slowly
- tape transfer rates ~25-30%/year?
- tape latencies improving only slowly
- robotic latencies improving only slowly
- (even DRAMs not getting faster fast enough)

Magnetic Disk Transfer Rate Trends



Roughly 25%/year improvement since 1989

Magnetic Disk Seek Time Trends



Roughly 10% improvement per year

Result:

There are more CPU cycles available to make every I/O decision - in some cases, **billions**.

You can view this as a **problem**, or as an **opportunity**.

Solutions

Parallelism:

- RAID -- better bandwidth, no latency help

Improved caching:

- better guesses by storage system

File system hints:

- great -- better information flow from application to system

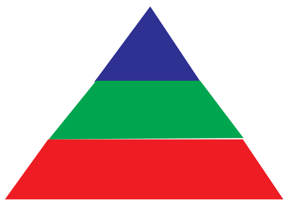
but:

- app can’t take advantage of **current state** of the storage system

SLEDs: Storage Latency Estimation Descriptors

- information flows from storage system to application
- estimate the time to start of data and rate it will arrive
- expose storage state to app in **hardware independent** fashion
- apps use to intelligently control access patterns
- integrate with reservations when necessary
- subsystems use to exchange information

Example: Satellite Image Database Scanning



RAM
disk
robotic tape

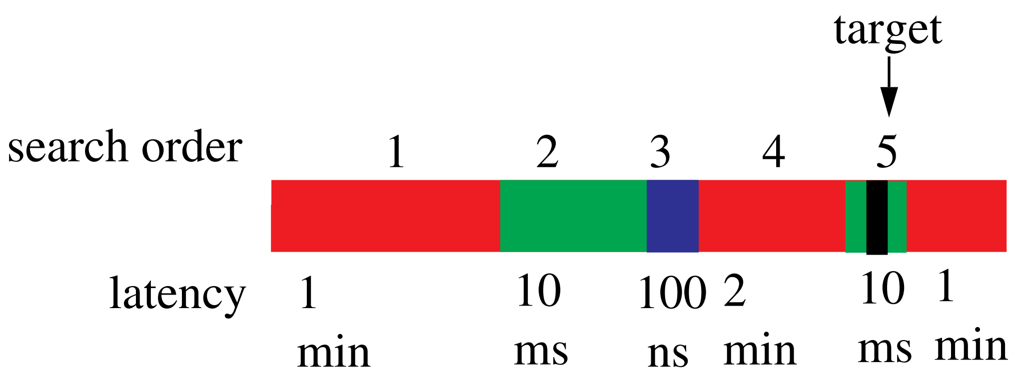
data migrates up and down the hierarchy on demand

data residency

initial state

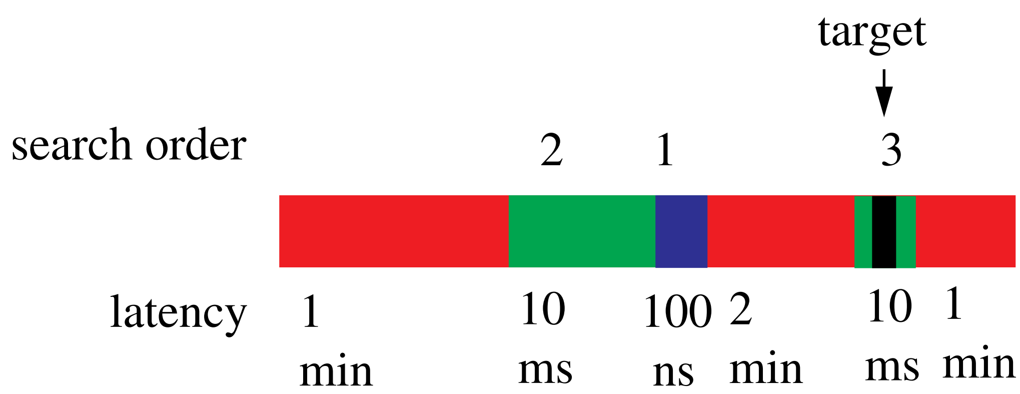
after significant use

Linear Search



without SLEDs: 3+ minutes to fetch target
scanning from beginning of dataset

Search Using SLEDs



using SLEDs: 20 milliseconds to fetch target due to intelligent request planning

4 orders of magnitude improvement!

Applications

- performance prediction for real-time apps
  (video on demand, interactive image database browsing)
- query optimization for databases
- I/O request pruning
- opportunistic piggybacking
  (take advantage of data others have requested)

Issues to be Addressed

- require algorithm changes
- require cooperation or modelling of subsystems
- distributed state hard to model or control
- effects of operations on future state hard to represent

Impact

- predictable performance means happier users
- intelligent access patterns lower I/O workload
- latency awareness reduces unnecessary and aborted requests
- device independence reduces legacy burden
- occasional “big win” multiple orders of magnitude runtime improvement

Conclusion

SLEDs will provide the "future-proof" interface that allows applications to intelligently manage their I/O workloads.

http://www.isi.edu/~rdv/sleds/