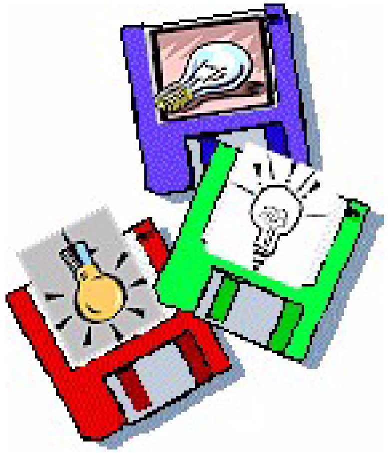


Active Disks - Remote Execution for Network-Attached Storage

Erik Riedel

Thesis Defense

Electrical and Computer Engineering

Prof. David Nagle, ECE
Prof. Christos Faloutsos, SCS
Prof. Garth Gibson, SCS
Prof. Pradeep Khosla, ECE
Dr. Jim Gray, Microsoft Research

Thesis Statement

A number of important I/O-intensive applications can take advantage of *computational power* available directly at storage devices

Computation in Storage

Thesis Statement

A number of important I/O-intensive applications can take advantage of *computational power* available directly at storage devices to improve their overall *performance*

Computation in Storage

Performance Model

Thesis Statement

A number of important I/O-intensive applications can take advantage of *computational power* available directly at storage devices to improve their overall *performance*, more effectively balance their consumption of system-wide *resources*

Computation in Storage

Performance Model

Applications & Prototype

Thesis Statement

A number of important I/O-intensive applications can take advantage of *computational power* available directly at storage devices to improve their overall *performance*, more effectively balance their consumption of system-wide *resources*, and provide *functionality* that would not otherwise be available.

Computation in Storage

Performance Model

Applications & Prototype

Drive-Specific Functionality

Outline

Motivation

Computation in Storage

Performance Model

Applications & Prototype

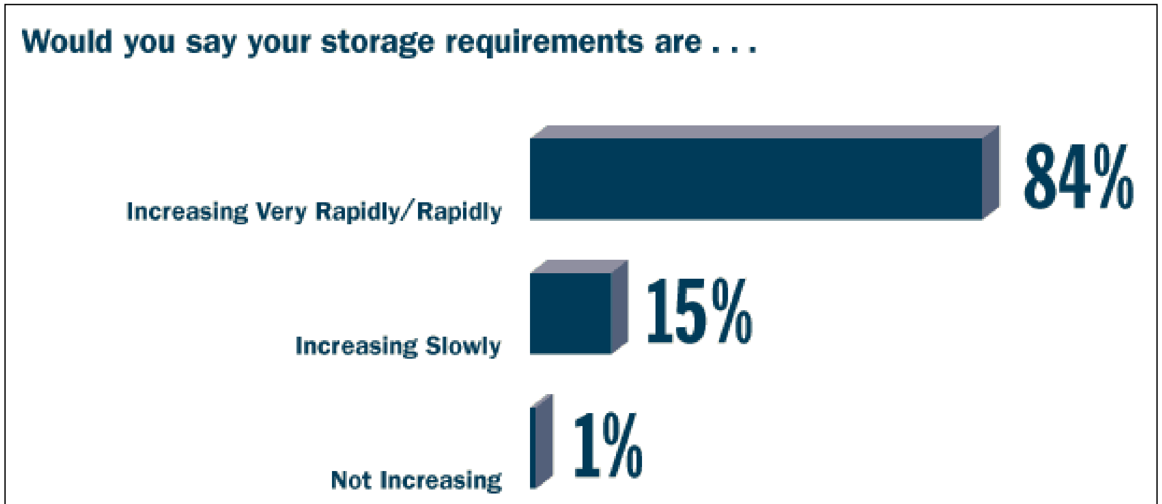
Drive-Specific Functionality

Related Work

Contributions & Future Work

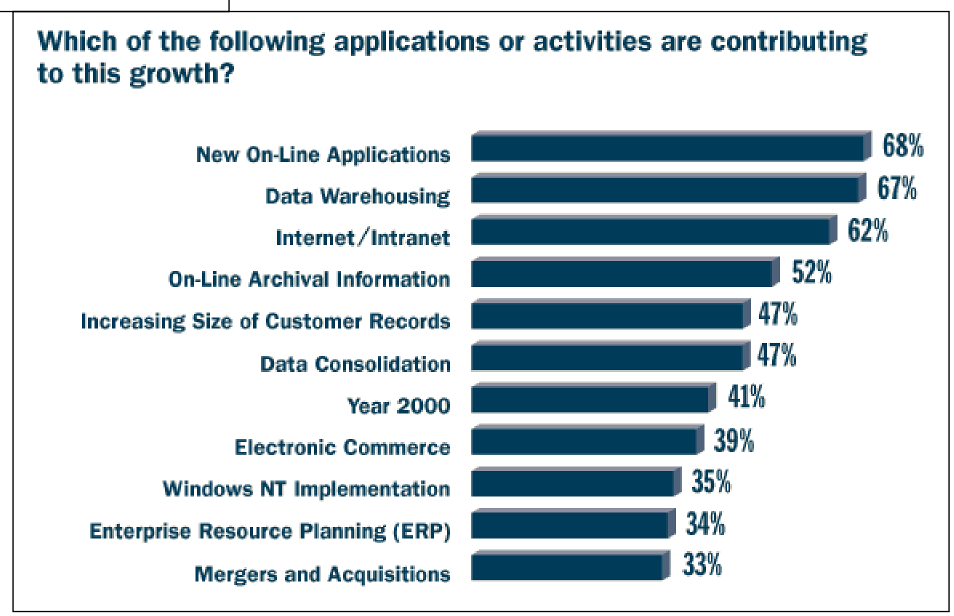
Motivation

Allow *faster, more flexible* access to storage



Storage requirements are pushing

- more data
- increased sharing
- richer data types
- novel applications



data from *www.EMC.com* survey of Senior IS Executives

Outline

Motivation

Computation in Storage

Performance Model

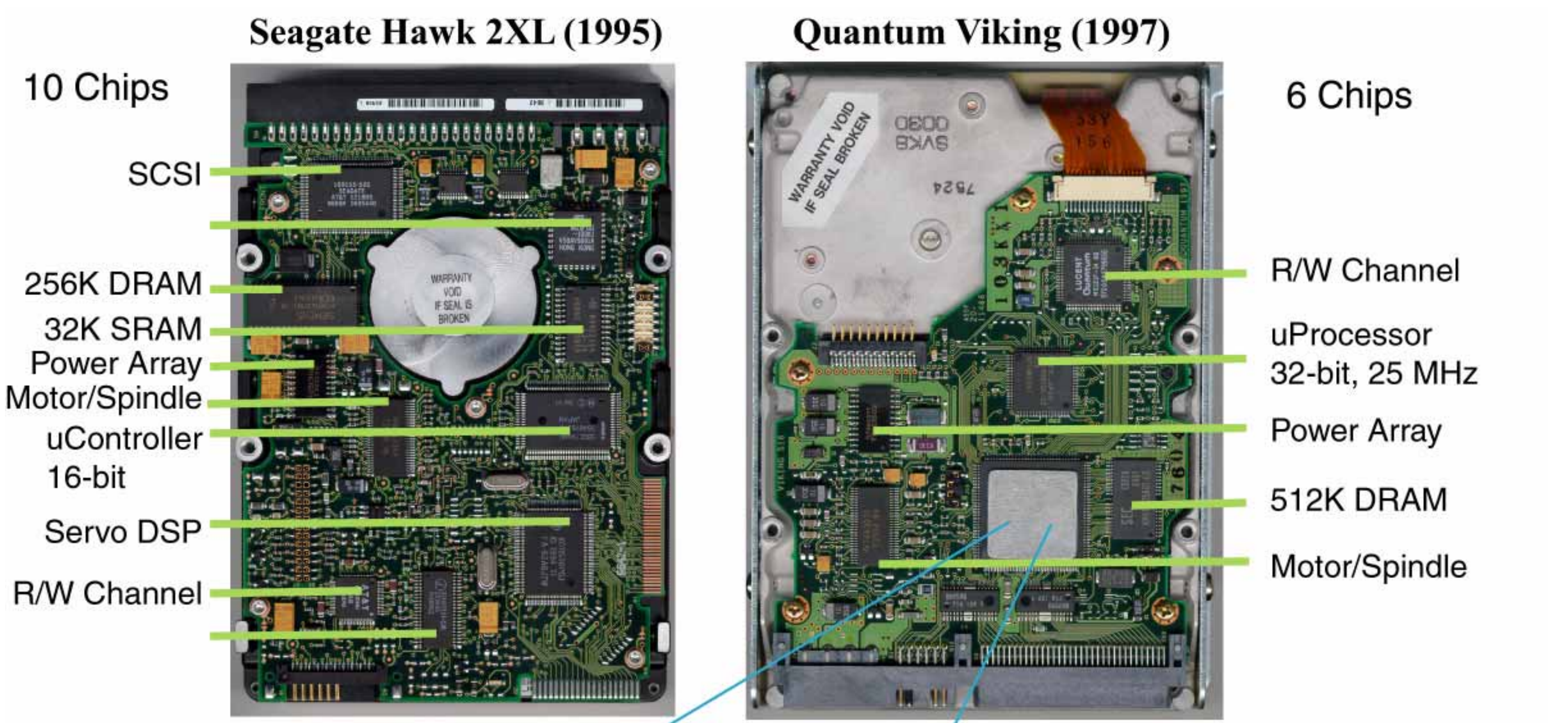
Applications & Prototype

Drive-Specific Functionality

Related Work

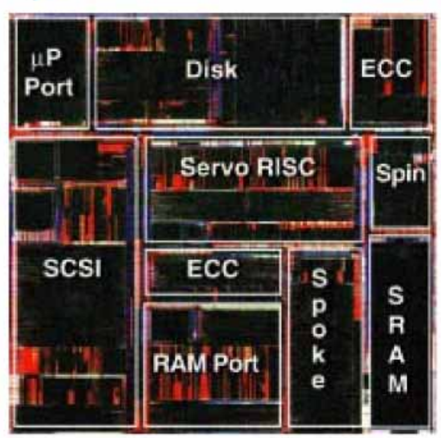
Contributions & Future Work

Evolution of Disk Drive Electronics

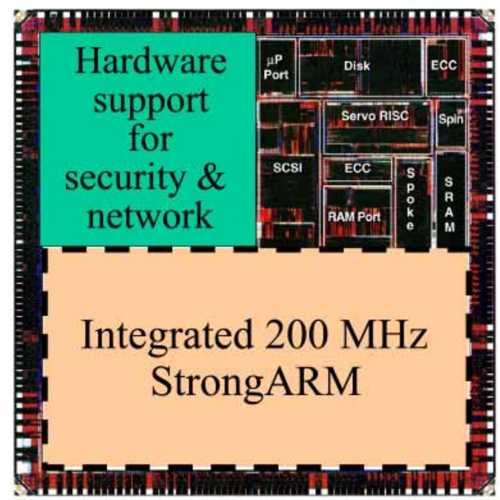


Integration

- reduces chip count
- improves reliability
- reduces cost
- future integration to processor on-chip
- but there must be at least *one* chip



Trident ASIC



Future Generation ASIC

Excess Device Cycles Are ~~Coming~~ Here

Higher and higher levels of integration in electronics

- specialized drive chips combined into single ASIC
- technology trends push toward integrated control processor
- Siemens TriCore - 100 MHz, 32-bit superscalar today
  - to 500 MIPS within 2 years, up to 2 MB on-chip memory
- Cirrus Logic 3CI - ARM7 core today
  - to ARM9 core at 200 MIPS in next generation

High volume, commodity product

- 145 million disk drives sold in 1998
  - about 725 petabytes of total storage
- manufacturers looking for value-added functionality

Opportunity

TPC-D 300 GB Benchmark, Decision Support System

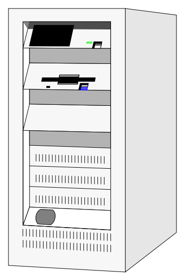
= 7,344 total MHz

Digital AlphaServer 8400
- 12 x 612 MHz 21164
- 8 GB memory
- 3 64-bit PCI busses
- 29 FWD SCSI controllers

3 x 266 = 798 MB/s

29 x 40 = 1,160 MB/s

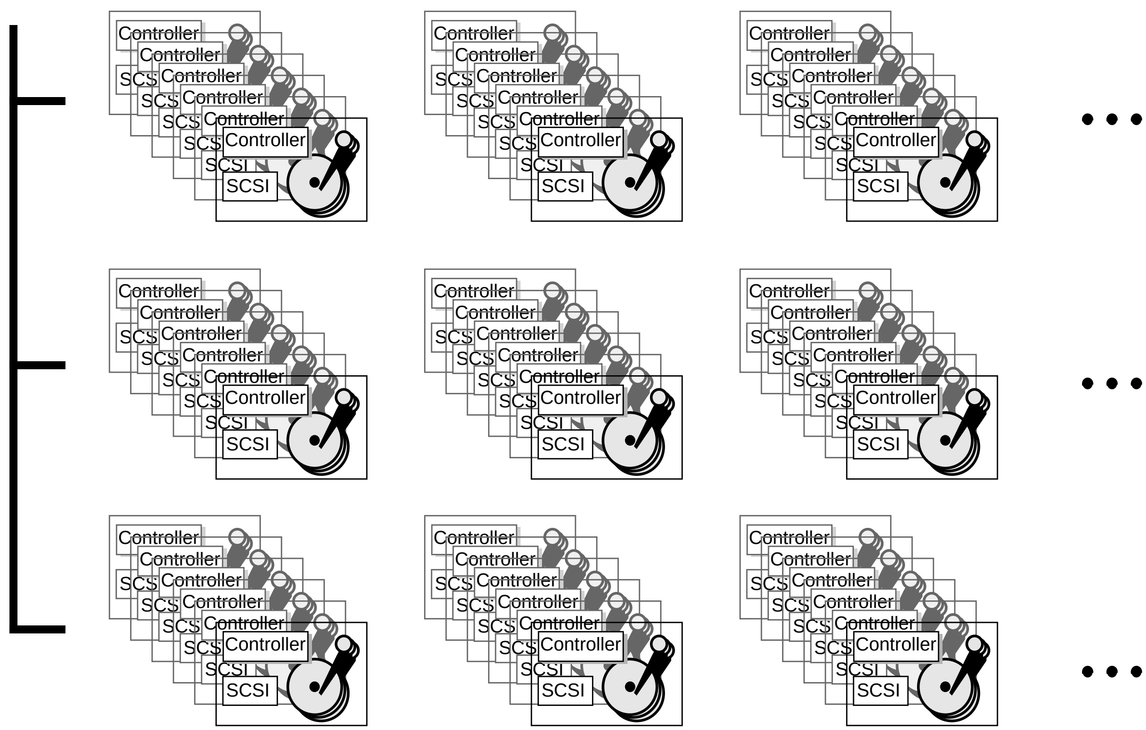
Database Server



Storage
- 520 rz29 disks
- 4.3 GB each
- 2.2 TB total

= 104,000 total MHz
(with 200 MHz drive chips)

= 5,200 total MB/s
(at 10 MB/s per disk)



Advantage - Active Disks

Active Disks **execute application-level code on drives**

Basic advantages of an Active Disk system

parallelism
- **parallel processing - lots of disks**

compute at the edges
- **bandwidth reduction - filtering operations are common**
- **scheduling - little bit of "strategy" can go a long way**

Characteristics of appropriate applications

- **execution time dominated by data-intensive "core"**
- **allows parallel implementation of "core"**
- **cycles per byte of data processed - *computation***
- **data reduction of processing - *selectivity***

Example Application

Data mining - association rules [Agrawal95]

- retail data, analysis of “shopping baskets”
- frequent sets summary counts
- count of *1-itemsets* and *2-itemsets*
- milk & bread => cheese
- diapers & beer

Partitioning with Active Disks

- each drive performs count of its portion of the data
- counts combined at host for final result

Outline

Motivation

Computation in Storage

Performance Model

Applications & Prototype

Drive-Specific Functionality

Related Work

Contributions & Future Work

Performance Model

Application Parameters

N_{in} = number of bytes processed

N_{out} = number of bytes produced

w = cycles per byte

t = run time for traditional system

t_{active} = run time for active disk system

d = number of disks

System Parameters

s_{cpu} = CPU speed of the host

r_d = disk raw read rate

r_n = disk interconnect rate

Active Disk Parameters

s_{cpu}' = CPU speed of the disk

r_d' = active disk raw read rate

r_n' = active disk interconnect rate

Traditional vs. Active Disk Ratios

$\alpha_N = N_{in}/N_{out}$ $\alpha_d = r_d'/r_d$ $\alpha_n = r_n'/r_n$ $\alpha_s = s_{cpu}'/s_{cpu}$

Performance Model

Traditional server:

$$t = max\left(\frac{N_{\text{in}}}{d \cdot r_d}, \frac{N_{\text{in}}}{r_n}, \frac{N_{\text{in}} \cdot w}{s_{\text{cpu}}}\right) + (1 - p) \cdot t_{serial}$$

disk network cpu overhead

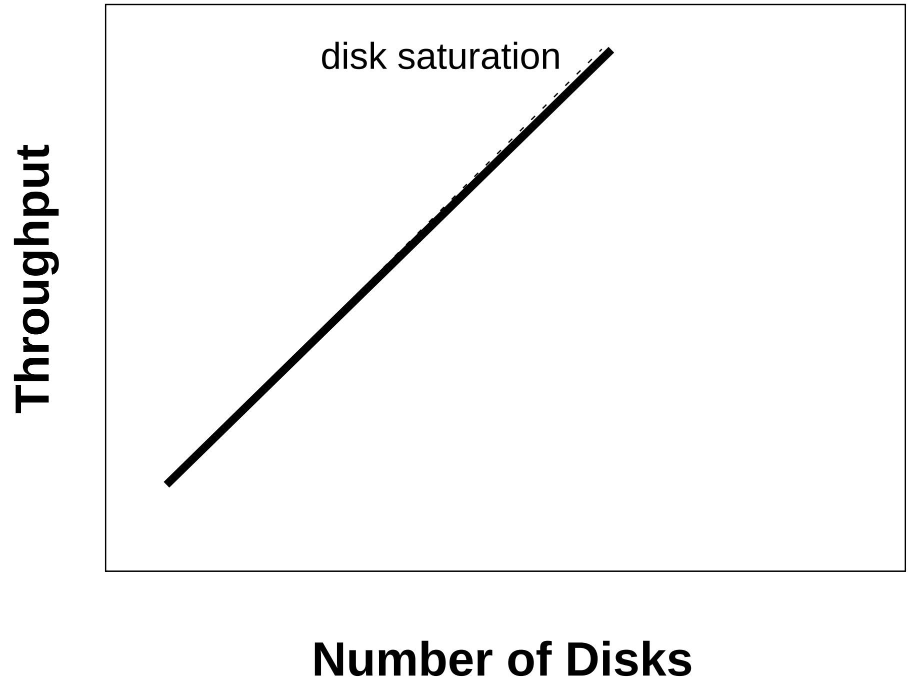
Active Disks:

$$t_{active} = max\left(\frac{N_{\text{in}}}{d \cdot r_d'}, \frac{N_{\text{out}}}{r_n'}, \frac{N_{\text{in}} \cdot w}{d \cdot s_{\text{cpu}}'}\right) + (1 - p) \cdot t_{serial}$$

Throughput Model

Scalable throughput

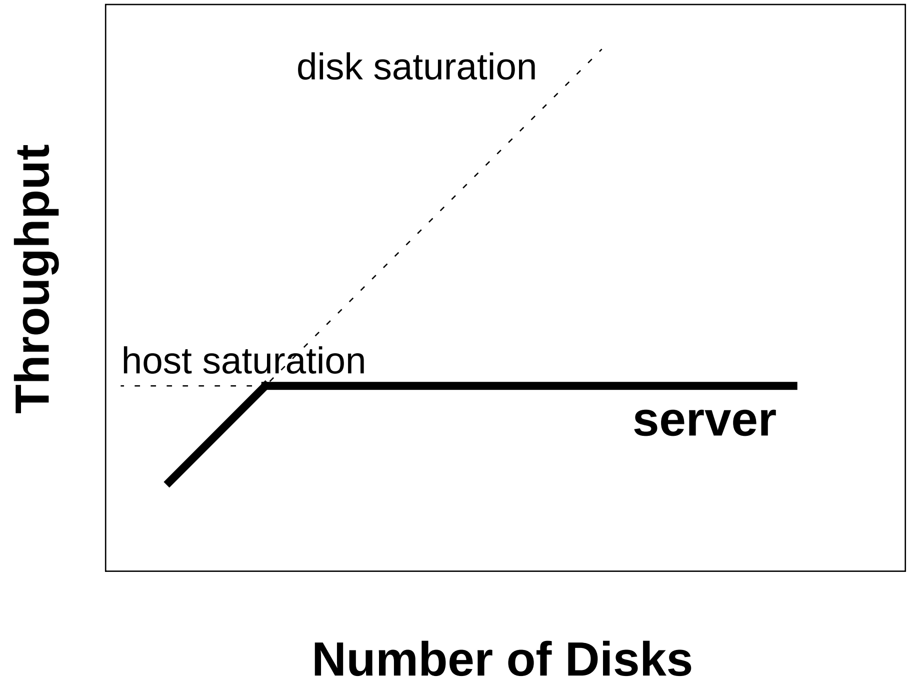
- **speedup** = (#disks)/(host-cpu-speed/disk-cpu-speed)



Throughput Model

Scalable throughput

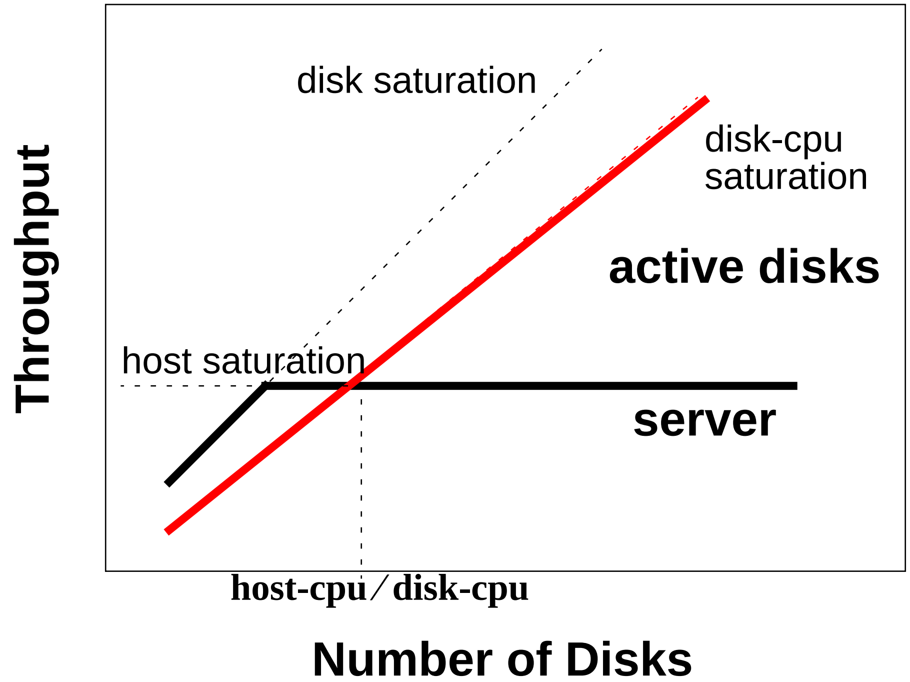
- **speedup = (#disks)/(host-cpu-speed/disk-cpu-speed)**



Throughput Model

Scalable throughput

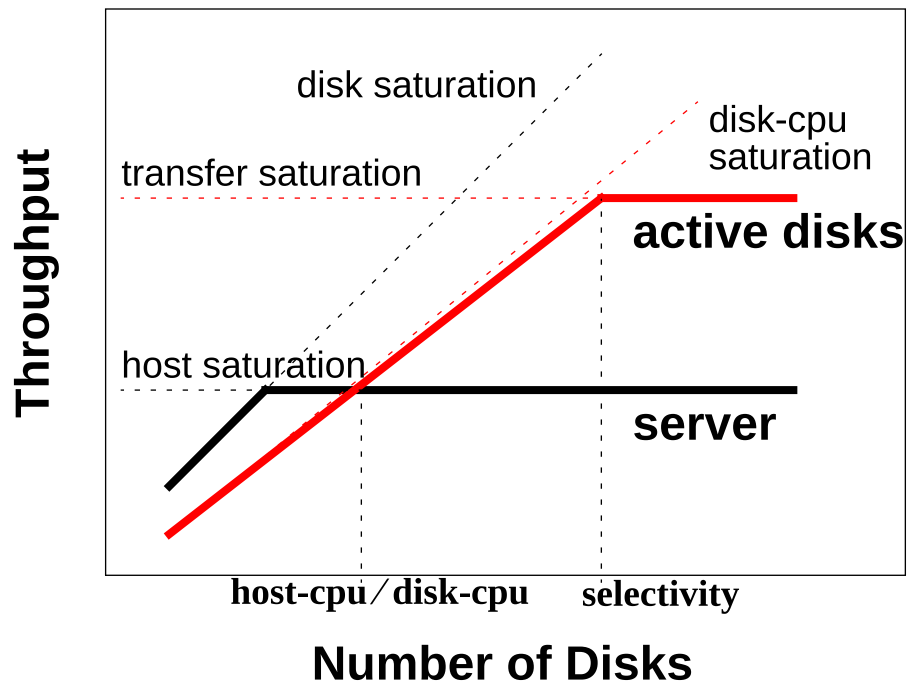
- speedup = (#disks)/(host-cpu-speed/disk-cpu-speed)
- (host-cpu/disk-cpu-speed) ~ 5   (two processor generations)



Throughput Model

Scalable throughput

- speedup = (#disks)/(host-cpu-speed/disk-cpu-speed)
- (host-cpu/disk-cpu-speed) ~ 5 (two processor generations)
- selectivity = #bytes-input / #bytes-output



Outline

Motivation

Computation in Storage

Performance Model

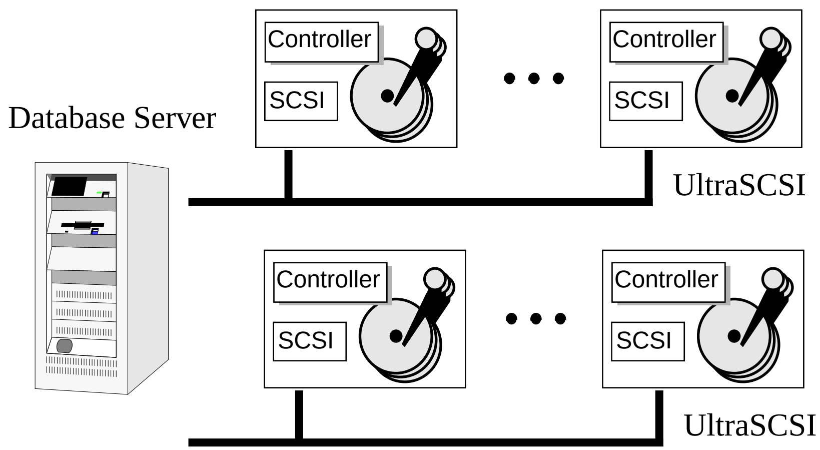
Applications & Prototype

Drive-Specific Functionality

Related Work

Contributions & Future Work

Prototype Comparison

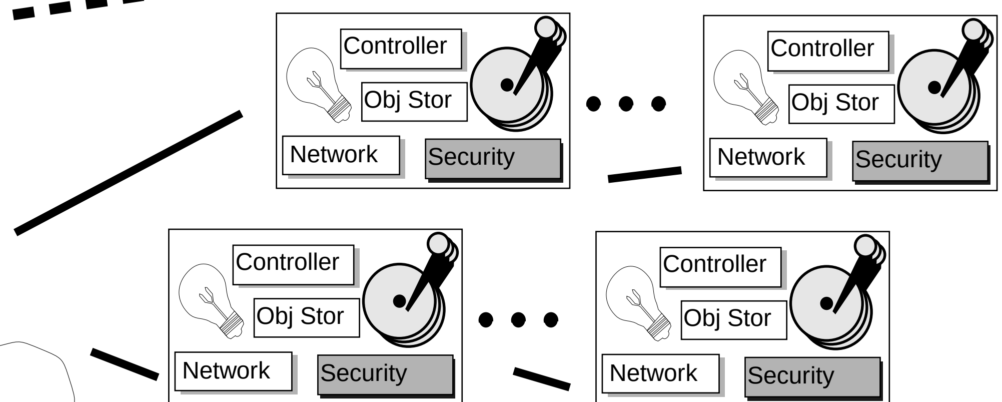


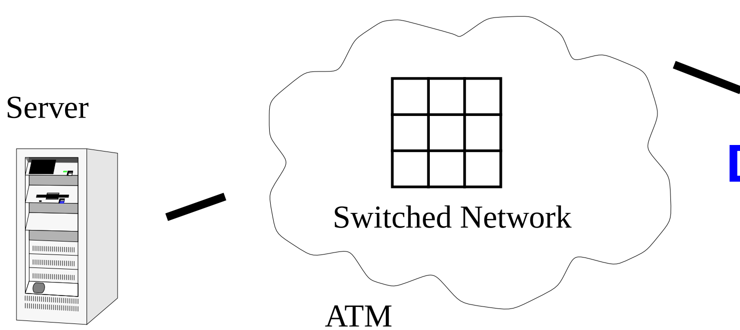
Traditional System

Digital AlphaServer 500/500

- **500 MHz, 256 MB memory**
- **Seagate Cheetah disks**
- **4.5 GB, 11.2 MB/s**

Active Disk System





Digital AXP 3000/400 “Active Disks”

- **133 MHz, 64 MB, software NASD**
- **Seagate Medallist disks**
- **4.1 GB, 6.5MB/s**

Data Mining & Multimedia

Data Mining - association rules [Agrawal95]

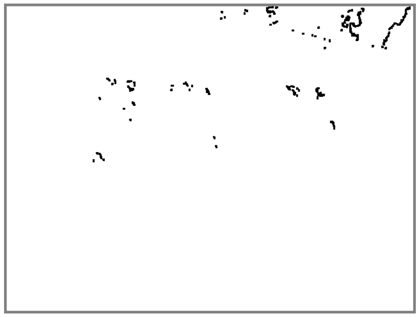
- frequent sets summary counts
- milk & bread => cheese

Database - nearest neighbor search

- *k* records closest to input record
- with large number of attributes, reduces to scan

Multimedia - edge detection [Smith95]

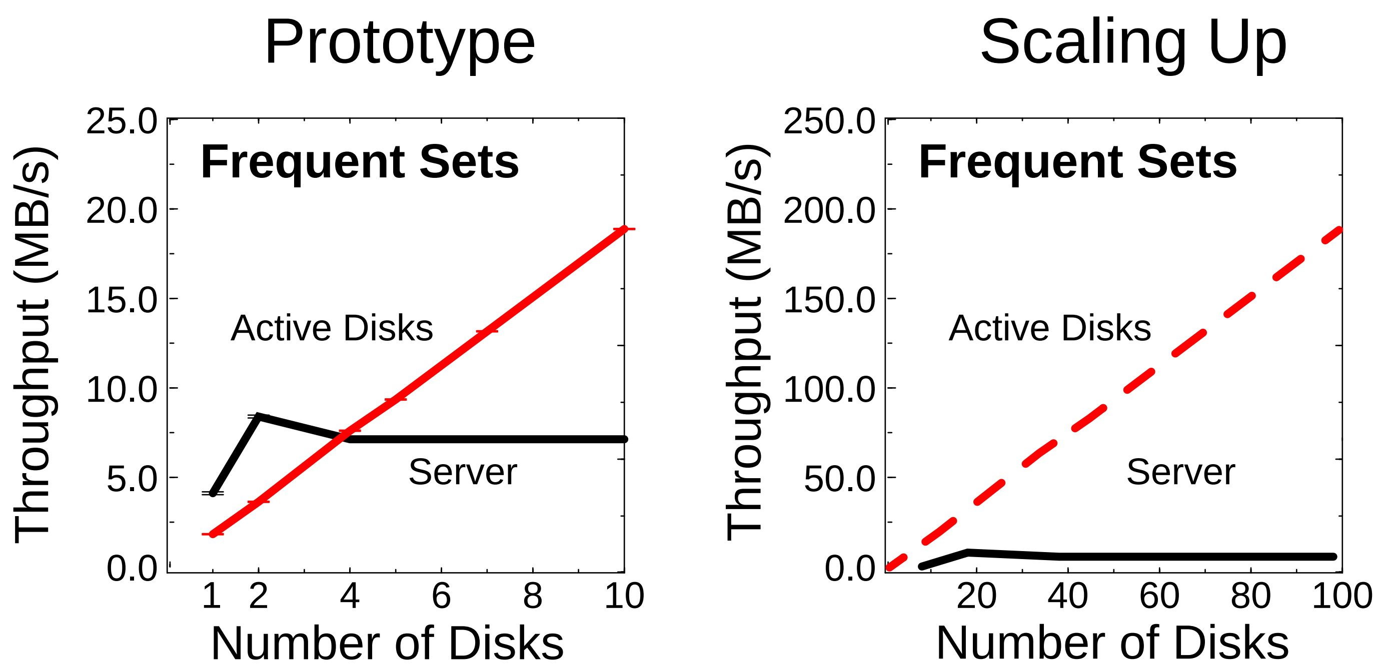
- detect edges in an image



Multimedia - image registration [Welling97]

- find rotation and translation from reference image

Data Mining & Multimedia

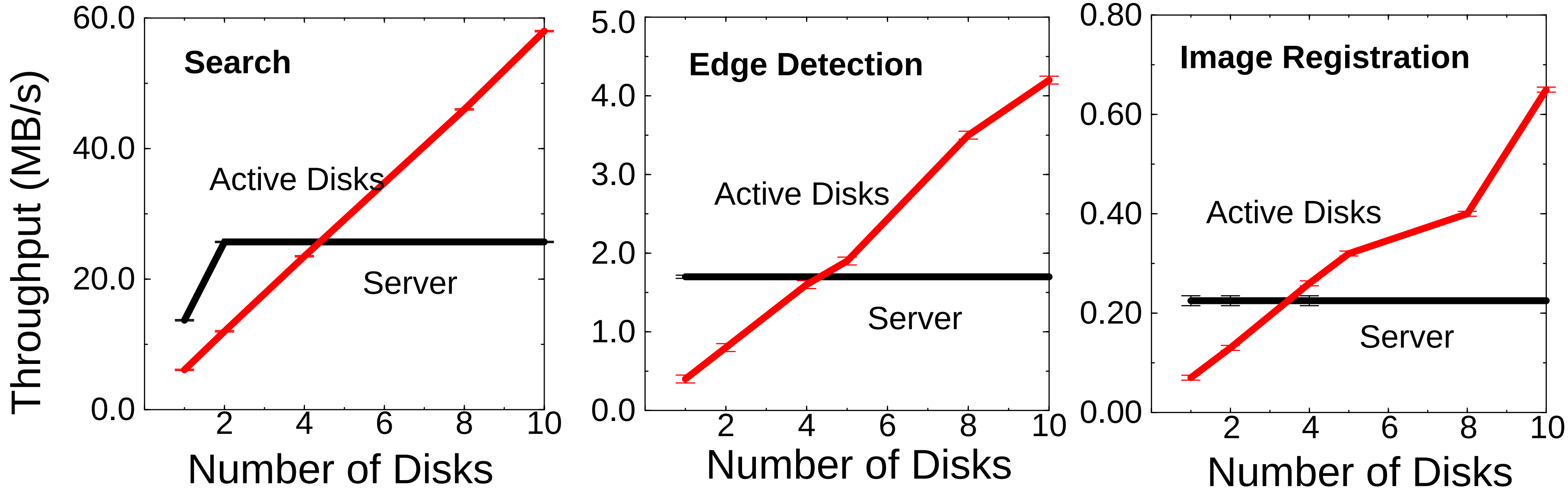


Prototype performance

- **factor of 2.5x with Active Disks**
- **scalable in a more realistic, larger system**

Performance with Active Disks

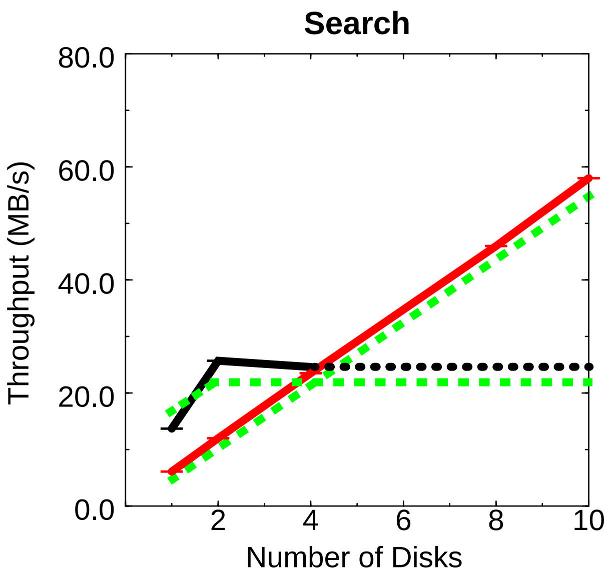
application	input	computation (inst/byte)	throughput (MB/s)	memory (KB)	selectivity (factor)	bandwidth (KB/s)
Search	k=10	7	28.6	72	80,500	0.4
Frequent Sets	s=0.25%	16	12.5	620	15,000	0.8
Edge Detection	t=75	303	0.67	1776	110	6.1
Image Registration	-	4740	0.04	672	180	0.2

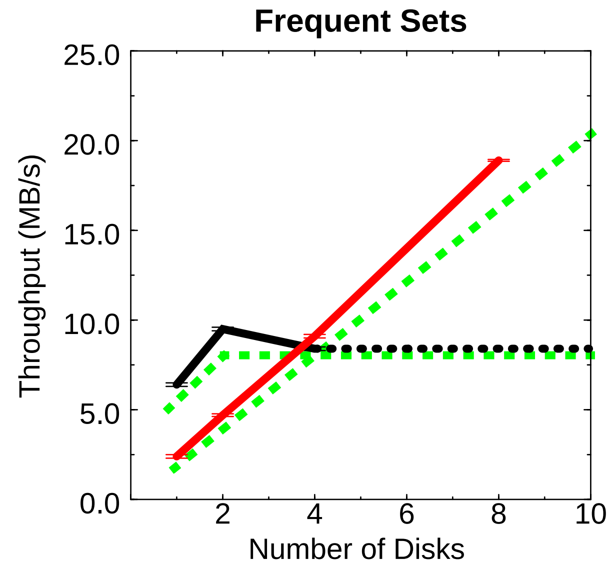


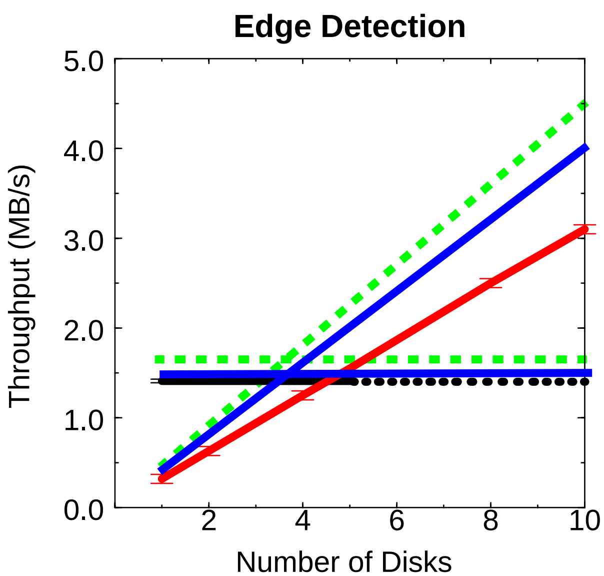
Scalable performance

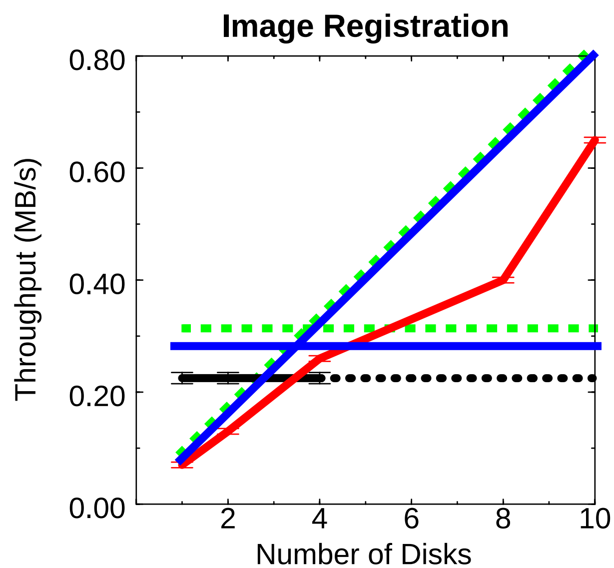
- **crossover at four disks - “technology gap”**
- **cycles/byte => throughput**
- **selectivity => network bottleneck**

Model Validation









Database Systems

Basic Operations

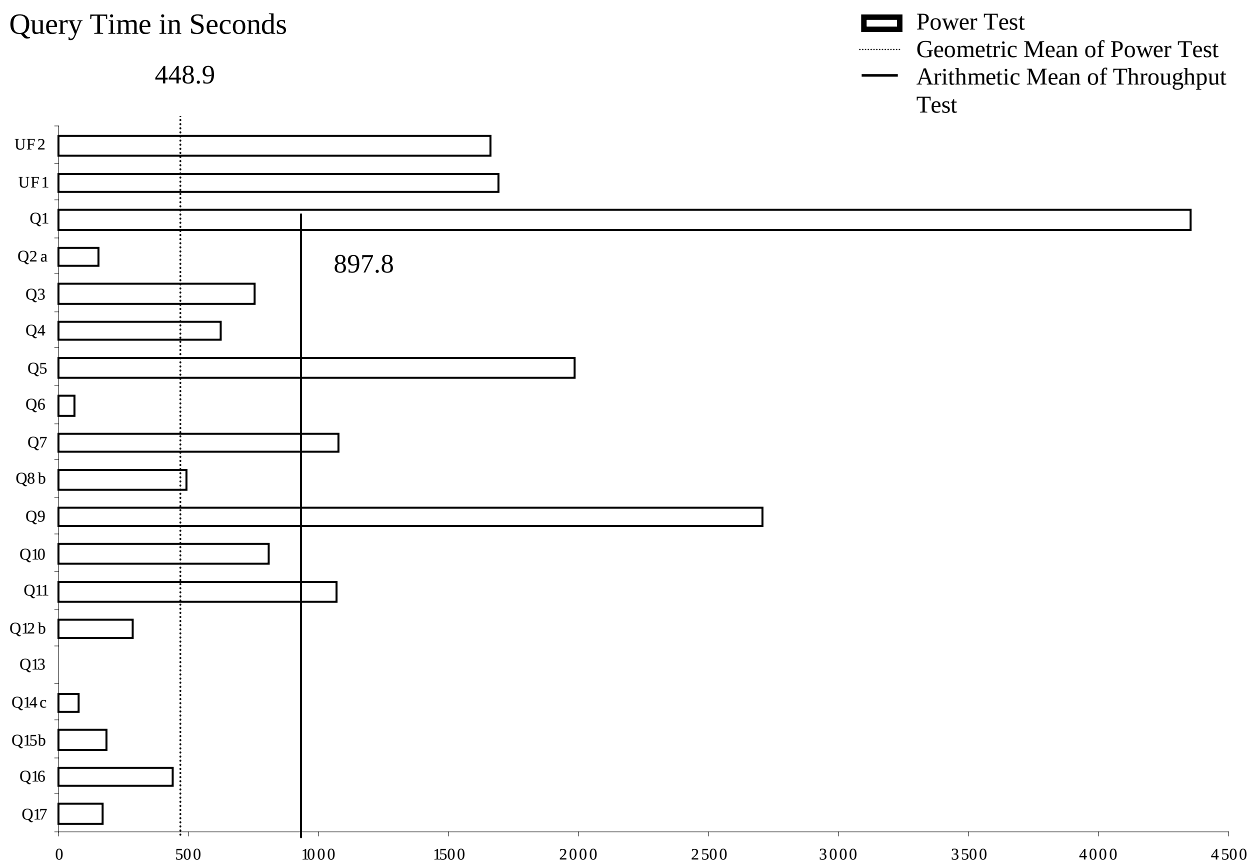
- select - scan
- project - scan & sort
- join - scan & hash-join

Workload

- TPC-D decision support
  - large data, scale factor of 300 GB uses 520 disks
  - ad-hoc queries
  - high-selectivity, “summary” questions

Digital Equipment Corporation & Oracle Corporation	Digital AlphaServer 8400 5/625 12 CPUs using Oracle8	TPC-D Rev. 1.3.1 Report Date: 27 May 98

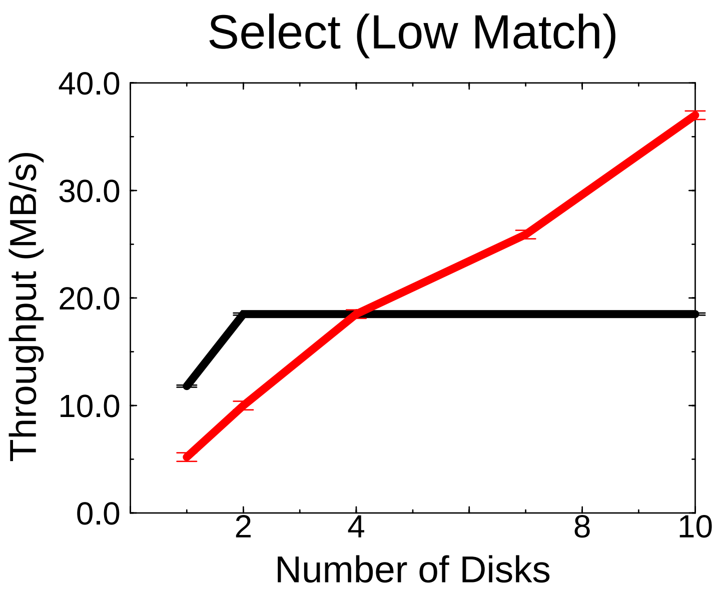
Total System Cost	TPC-D Power	TPC-D Throughput	Price/Performance	
$2,649,262	**2406.2** QppD@ 300GB	**986.1** QthD@ 300GB	**$1,720** QphD@ 300GB	
Database Size	Database Manager	Operating System	Other Software	Availability Date
300GB	**Oracle8 v8.0.4**	**Digital UNIX V4.0D**	**None**	**May 27, 1998**



Database Load Time = 22 hours 50 minutes 17 seconds   Disk Size/Database Size=7.47   RAID: No

Components	**Qty**	**Type**
Processors	12	612 Mhz DECchip 21164
Cache Memory per Processor		4MB
Memory	2	4 GB
Disk Controllers	29	PCI
Disks	521	4.3 GB Disks
Total Disk Storage		2240.3GB

Active PostgreSQL Select



Experimental setup

- database is PostgreSQL 6.5
- server is 500 MHz Alpha, 256 MB
- disks are Seagate Cheetahs
- vs. *n* Active Disks
  - 133 MHz Alpha, 64 MB
  - Digital UNIX 3.2g
- ATM networking vs. Ultra SCSI

performance results

- SQL `select` operation (selectivity = 52)
- interconnect limited
- scalable Active Disk performance

Database - Aggregation (Project)

l_return	sum_revenue	sum_qty
A	39599.7	29
R	67936.6	71

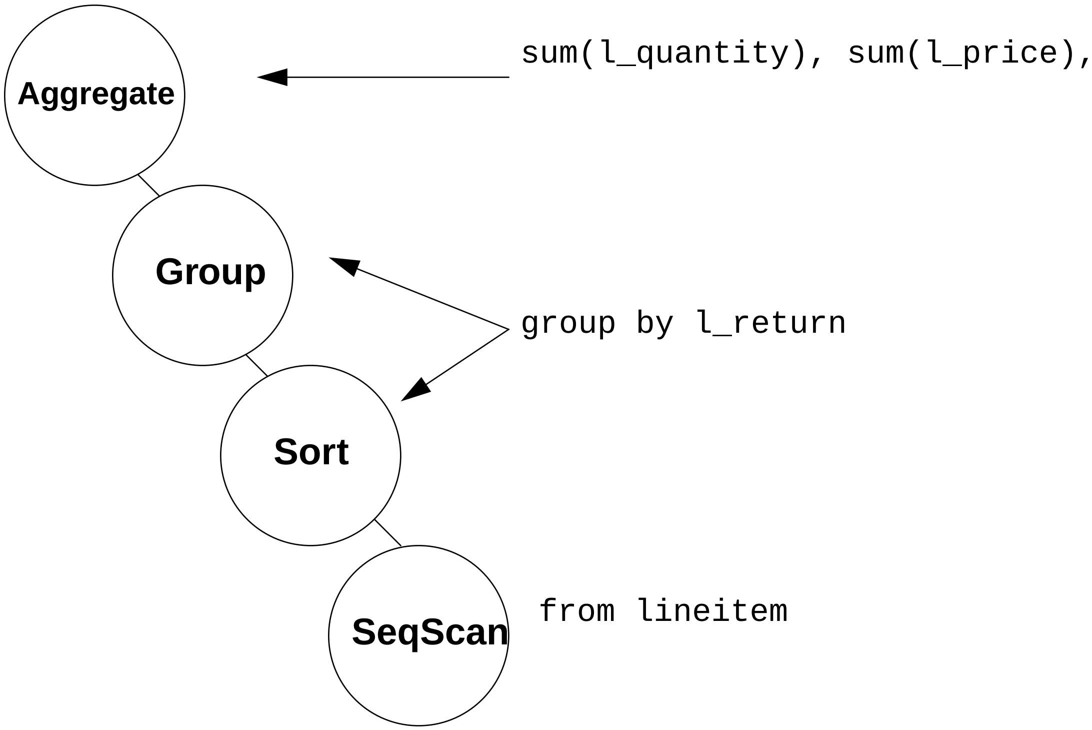
```
select sum(l_price), sum(l_qty)
from lineitem
group by l_return
```

relation S

l_orderkey	l_shipdate	l_qty	l_price	l_return
1730	01-25-93	6	11051.6	A
3713	04-12-96	32	29600.3	R
7010	10-05-98	23	29356.3	A
...				
32742	05-05-95	8	9281.9	R
36070	11-27-98	31	34167.9	R

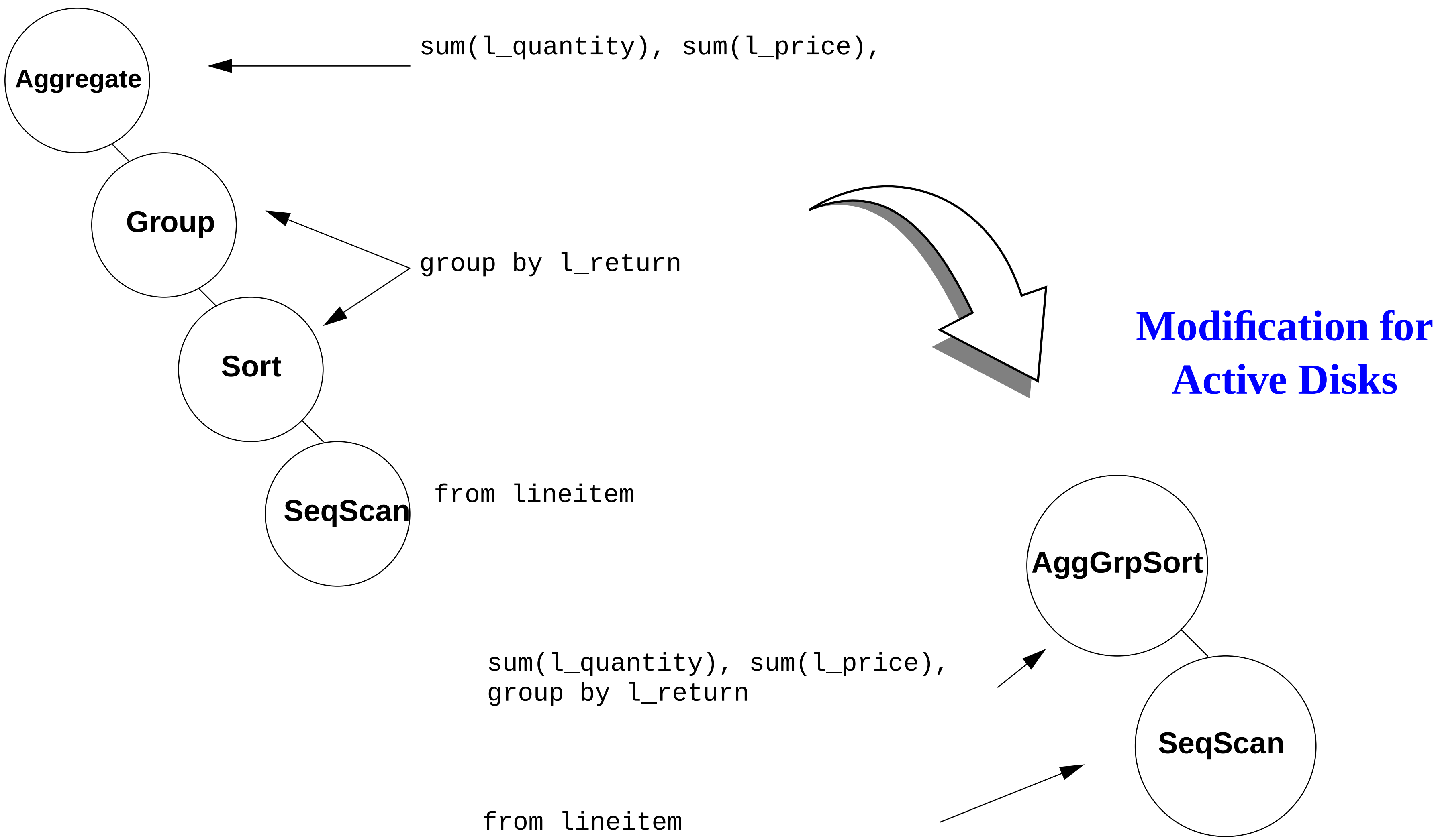
Database - Aggregation II

Query Plan   Query Text

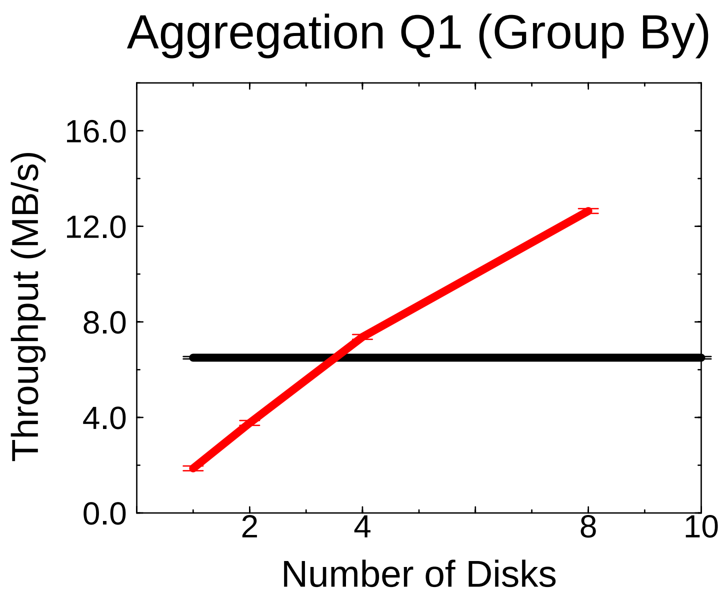


Database - Aggregation II

Query Plan   Query Text



Active PostgreSQL Aggregation



Algorithm

- **replacement selection sort**
- **maintain sorted heap in memory**
- **combine (aggregate) records when keys match exactly**

Benefits

- **memory requirements determined by *output* size**
- **longer average run length**
- **easy to make *adaptive***

Disadvantage

- **poor memory behavior vs. qsort**

performance results

- **SQL `sum()...group by` operation (selectivity = 650)**
- **cycles/byte = 32, cpu limited**

Database - Join

l_return	sum_revenue	sum_qty
A	40407.9	29
R	34167.9	31

```
select sum(l_price), sum(l_qty)
from lineitem, part
where p_name like '%green%'
and l_partkey = p_partkey
group by l_return
```

relation R

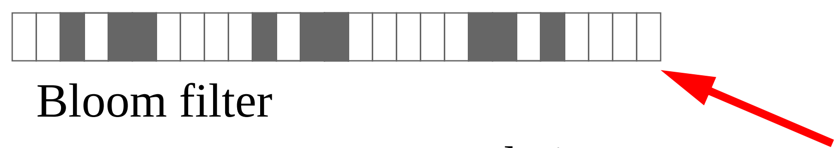
p_partkey	p_name	p_brand	p_type
2593	green car	vw	11
5059	red boat	fast	29
1098	green tree	pine	35
...			
0412	blue sky	clear	92
5692	red river	dirty	34

relation S

l_orderkey	l_partkey	l_qty	l_price	l_return
1730	2593	6	11051.6	A
3713	0412	32	29600.3	R
7010	1098	23	29356.3	A
...				
32742	5059	8	9281.9	R
36070	2593	31	34167.9	R

Bloom Join

```
select sum(l_price), sum(l_qty)
from lineitem, part
where p_name like '%green%'
and l_partkey = p_partkey
group by l_return
```



Bloom filter

relation R

p_partkey	p_name	p_brand	p_type
2593	green car	vw	11
5059	red boat	fast	29
1098	green tree	pine	35
0412	blue sky	clear	92
5692	red river	dirty	34

Bloom Join

```
select sum(l_price), sum(l_qty)
from lineitem, part
where p_name like '%green%'
and l_partkey = p_partkey
group by l_return
```

l_orderkey	l_partkey	l_qty	l_price	l_return
1730	2593	6	11051.6	A
7010	1098	23	29356.3	A

...

36070	2593	31	34167.9	R

relation S

l_orderkey	l_partkey	l_qty	l_price	l_return
1730	2593	6	11051.6	A
3713	0412	32	29600.3	R
7010	1098	23	29356.3	A

...

32742	5059	8	9281.9	R
36070	2593	31	34167.9	R

Bloom filter

relation R

p_partkey	p_name	p_brand	p_type
2593	green car	vw	11
5059	red boat	fast	29
1098	green tree	pine	35

...

0412	blue sky	clear	92
5692	red river	dirty	34

Bloom Join

```
select sum(l_price), sum(l_qty)
from lineitem, part
where p_name like '%green%'
and l_partkey = p_partkey
group by l_return
```

l_orderkey	l_partkey	l_qty	l_price	l_return
1730	2593	6	11051.6	A
7010	1098	23	29356.3	A
...				
36070	2593	31	34167.9	R

relation S

l_orderkey	l_partkey	l_qty	l_price	l_return
1730	2593	6	11051.6	A
3713	0412	32	29600.3	R
7010	1098	23	29356.3	A
...				
32742	5059	8	9281.9	R
36070	2593	31	34167.9	R

Bloom filter

relation R

p_partkey	p_name	p_brand	p_type
2593	green car	vw	11
5059	red boat	fast	29
1098	green tree	pine	35
...			
0412	blue sky	clear	92
5692	red river	dirty	34

Bloom Join

l_return	sum_revenue	sum_qty
A	40407.9	29
R	34167.9	31

l_orderkey	l_partkey	l_qty	l_price	l_return
1730	2593	6	11051.6	A
7010	1098	23	29356.3	A
...				
36070	2593	31	34167.9	R

```
select sum(l_price), sum(l_qty)
from lineitem, part
where p_name like '%green%'
and l_partkey = p_partkey
group by l_return
```

relation S

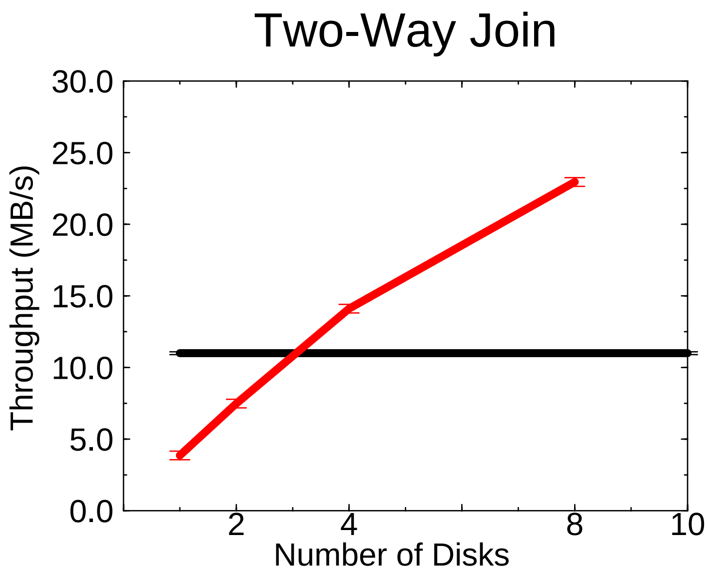
l_orderkey	l_partkey	l_qty	l_price	l_return
1730	2593	6	11051.6	A
3713	0412	32	29600.3	R
7010	1098	23	29356.3	A
...				
32742	5059	8	9281.9	R
36070	2593	31	34167.9	R

Bloom filter

relation R

p_partkey	p_name	p_brand	p_type
2593	green car	vw	11
5059	red boat	fast	29
1098	green tree	pine	35
...			
0412	blue sky	clear	92
5692	red river	dirty	34

Active PostgreSQL Join



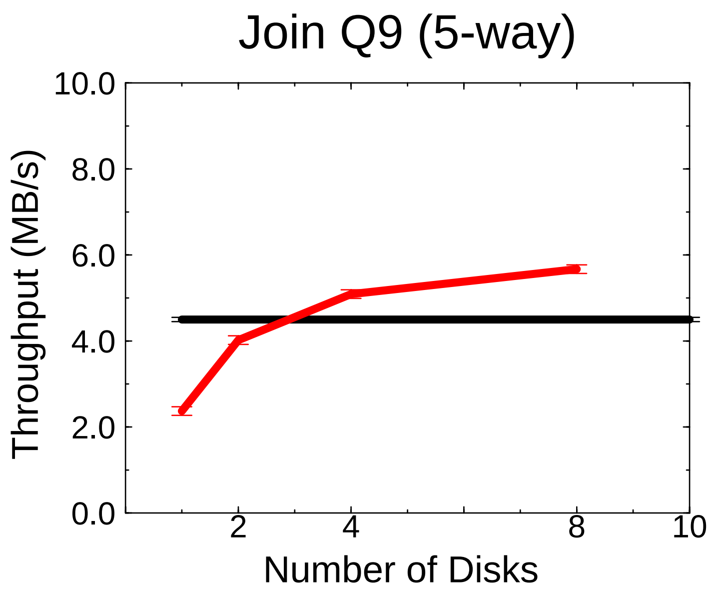
Algorithm

- read R to host
- create hash table for R
  - generate Bloom filter
- broadcast filter to all disks
- parallel scan at disks
  - semi-join to host
- final join at host

performance results

- SQL `2-way join` operation (selectivity = 8)
- will eventually be network limited

Active PostgreSQL Join II



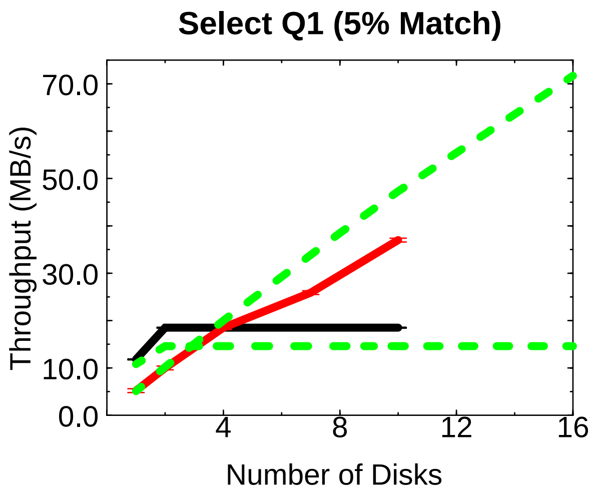
Experimental setup

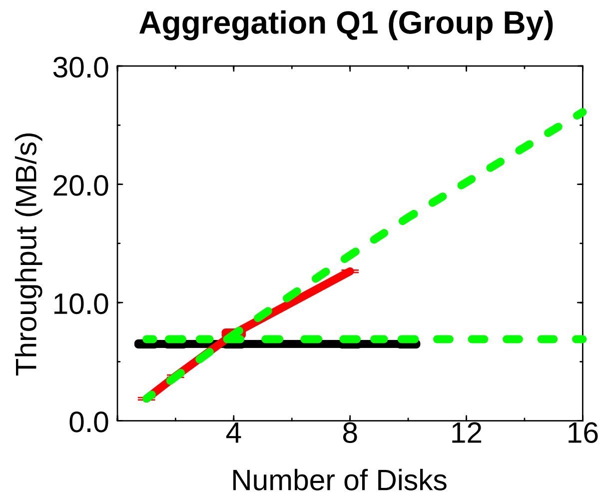
- **database is PostgreSQL 6.5**
- **server is 500 MHz Alpha, 256 MB**
- **disks are Seagate Cheetahs**
- **vs. *n* Active Disks**
  - **133 MHz Alpha, 64 MB**
  - **Digital UNIX 3.2g**
- **ATM networking vs. Ultra SCSI**

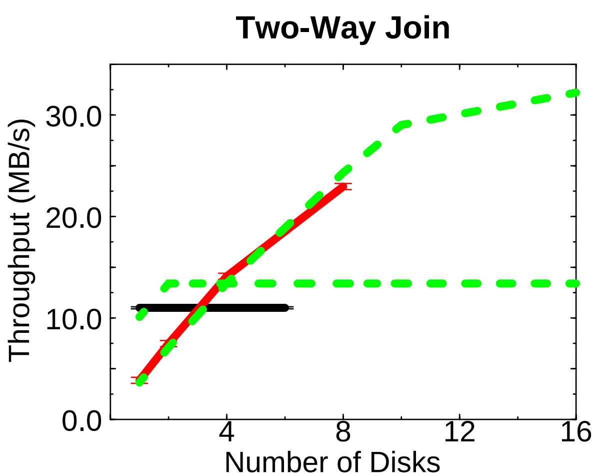
performance results

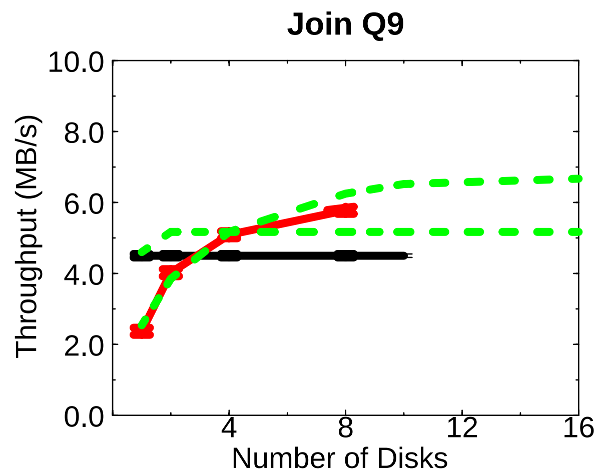
- **SQL `5-way join` operation**
- **large serial fraction, Amdahl's Law kicks in**

Model Validation (Database)









Database - Summary

Active PostgreSQL Prototype

Query	Bottleneck	Traditional (seconds)	Active Disks (seconds)	Improvement
Q1	computation	76.0	38.0	**100%**
Q5	serial fraction	219.0	186.5	**17%**
Q6	interconnect	27.2	17.0	**60%**
Q9	serial fraction	95.0	85.4	**11%**

Measured performance

- **four most expensive of the 17 TPC-D queries**
- **compares eight disk systems**
- **PostgreSQL 6.5 with Active Disk modifications**

Database - Extrapolation

Estimated Speedup on Digital 8400 (TPC-D, May 1998)

Query	Bottleneck	Traditional (seconds)	Active Disks (seconds)	Improvement
Q1	computation	4,357.1	*307.7*	***1,320%***
Q5	serial fraction	1988.2	*1,470.8*	***35%***
Q6	interconnect	63.1	*6.1*	***900%***
Q9	serial fraction	2710.8	*2,232.1*	***22%***

Predicted performance

- **comparison of Digital 8400 with 520 traditional disks**
- **vs. the same system with 520 Active Disks**

Database - Extrapolation

Estimated Speedup on Digital 8400 (TPC-D, May 1998)

Query	**Bottleneck**	**Traditional** (seconds)	**Active Disks** (seconds)	**Improvement**
Q1	computation	4,357.1	*307.7*	***1,320%***
Q5	serial fraction	1988.2	*1,470.8*	***35%***
Q6	interconnect	63.1	*6.1*	***900%***
Q9	serial fraction	2710.8	*2,232.1*	***22%***
Other Qs		*assume unchanged*		
Overall		18,619.5	*13,517.0*	***38%***

Predicted performance

- **comparison of Digital 8400 with 520 traditional disks**
- **vs. the same system with 520 Active Disks**

Database - Extrapolation

Estimated Speedup on Digital 8400 (TPC-D, May 1998)

Query	Bottleneck	Traditional (seconds)	Active Disks (seconds)	Improvement
Q1	computation	4,357.1	*307.7*	*1,320%*
Q5	serial fraction	1988.2	*1,470.8*	*35%*
Q6	interconnect	63.1	*6.1*	*900%*
Q9	serial fraction	2710.8	*2,232.1*	*22%*
Other Qs		*assume unchanged*		
Overall		18,619.5	*13,517.0*	*38%*
Cost		$2,649,262	*$3,034,045*	*15%*

Predicted performance

- **comparison of Digital 8400 with 520 traditional disks**
- **vs. the same system with 520 Active Disks**
- **overall cost increase of about 15%**
  - **assuming an Active Disk costs *twice* a traditional disk**

Outline

Motivation

Computation in Storage

Performance Model

Applications & Prototype

Drive-Specific Functionality

Related Work

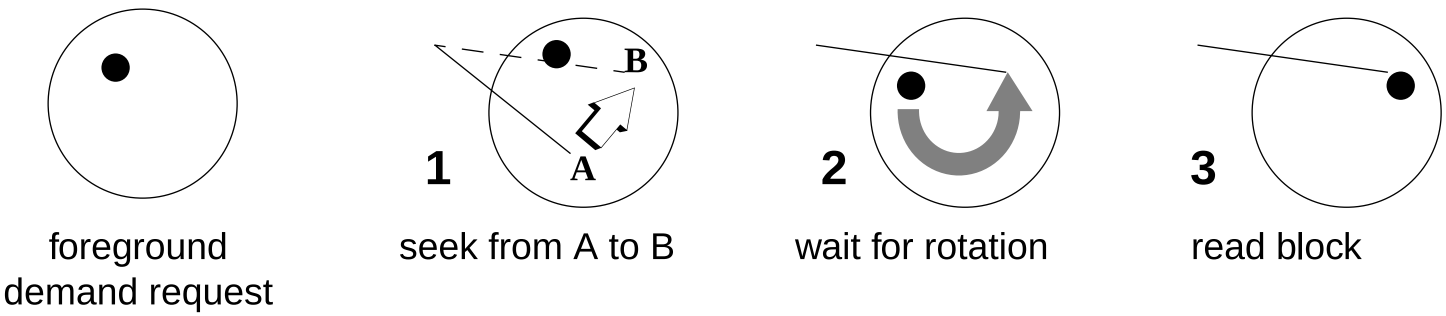
Contributions & Future Work

Additional Functionality

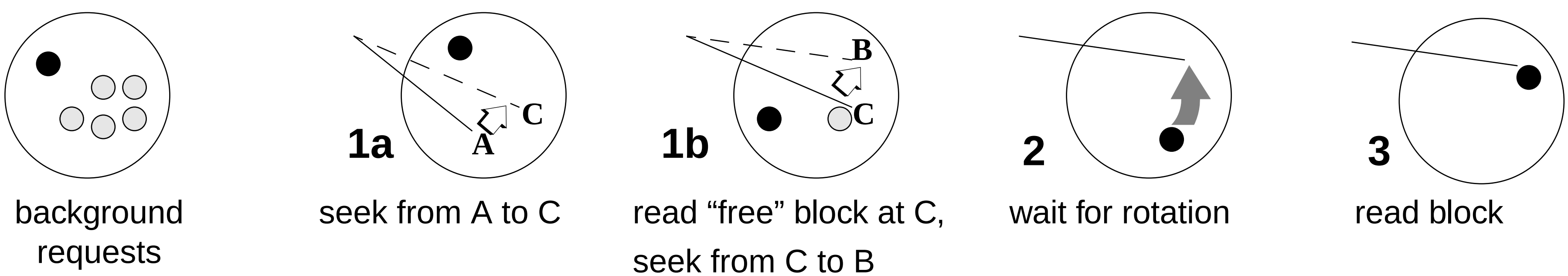
Data Mining for Free

- process sequential workload during “idle” time in OLTP
- allows e.g. data mining on an OLTP system

Action in Today’s Disk Drive

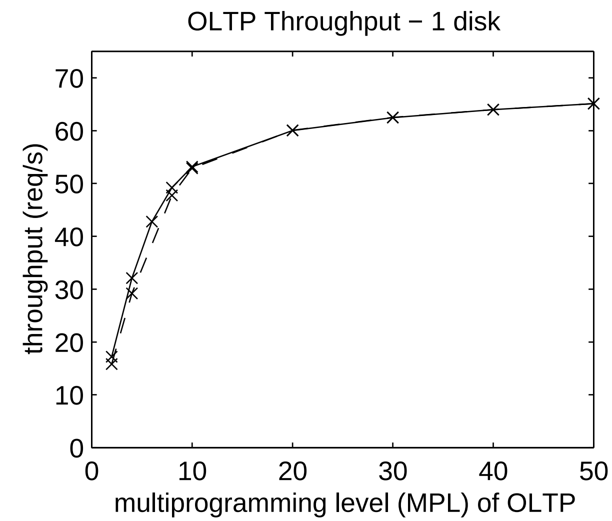


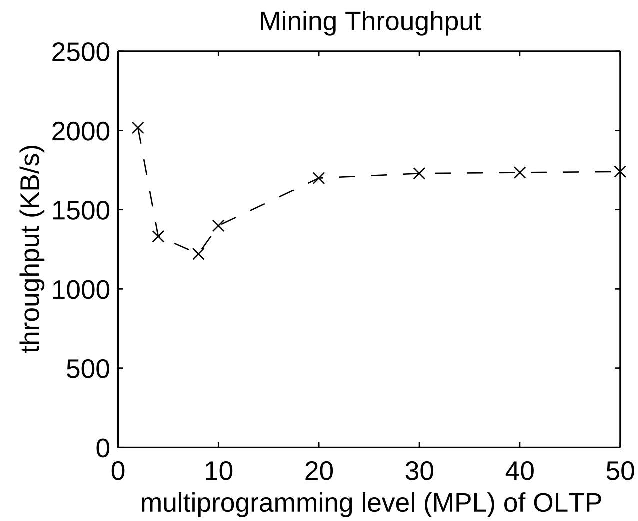
Modified Action With “Free” Block Scheduling

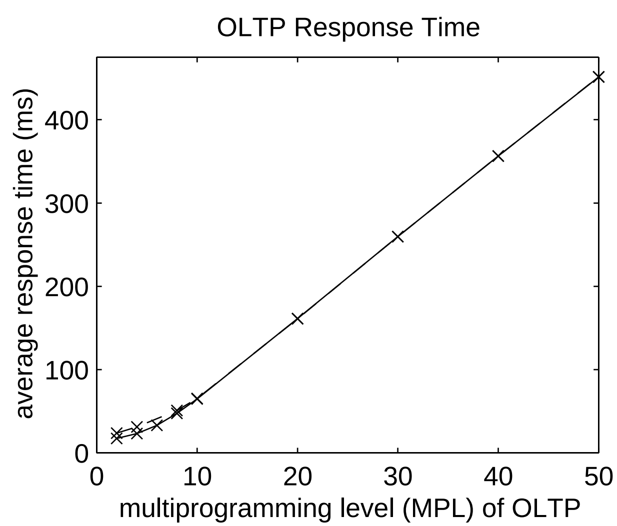


Data Mining for Free

- **combine background and “free” blocks**







Integrated scheduling

- **possible only at drives**
- **combines application-level and disk-level information**
- **achieves 30% of the drives sequential band-width “for free”**

Outline

Motivation

Computation in Storage

Performance Model

Applications & Prototype

Drive-Specific Functionality

Related Work

Contributions & Future Work

Related Work

Database Machines (CASSM, RAP, Gamma)

- today's advantages - higher disk bandwidth, parallelism
- general-purpose programmability
- parallel databases (Teradata, Tandem, Oracle, IBM)
- CAFS and SCAFS search accelerator (ICL, Fujitsu)

Parallel Programming

- automatic data parallelism (HPF), task parallelism (Fx)
- parallel I/O (Kotz, IBM, Intel)

Parallel Database Operations

- scan [Su75, Ozkarahan75, DeWitt81, ...]
- sort [Knuth73, Salzberg90, DeWitt91, Blelloch97, ...]
- hash-join [Kitsuregawa83, DeWitt85, ...]

Related Work - “Smart Disks”

Intelligent Disks (Berkeley)

- SMP database functions [Keeton98]
- analytic model, large speedups for join and sort (!)
- different architecture - *everything* is iDisks
- disk layout [Wang98], write optimizations

Programming Model (Santa Barbara/Maryland)

- select, sort, image processing via extended SCSI [Acharya98]
- simulation comparisons among Active Disks, Clusters, SMPs
- focus on network bottlenecks

SmartSTOR (Berkeley/IBM)

- analysis of TPC-D, significant benefits possible (!)
- suggest using one processor for multiple disks
- “simple” functions have limited benefits

Contributions

Exploit technology trends

- "excess" cycles on individual disk drives
- large systems => lots of disks => lots of power

Analytic

- performance model - predicts within 25%
- algorithms & query optimizer - map to Active Disk functions

Prototype

- data mining & multimedia
  - 2.5x in prototype, scale to 10x
- database with TPC-D benchmark
  - 20% to 2.5x in prototype, extrapolate 35% to 15x in larger system
- changed ~2% of database code, run ~5% of code at drives

Novel functionality

- data mining for free - close to 30% bandwidth "for free"

Conclusion - lots of potential *and* realistically attainable

Future Work

Extension of Database Functions

- optimization for index-based scans
- update and small request performance

Programming Model - Application Layers

- explicit programmer-controlled?
- vs. fully adaptive application mobility?
- databases have query optimizers, filesystems don't
- challenges: identify "structure" and identify "functions"

Masses of Storage, Pervasive Storage

- large volumes of data
- really large scale (1,000s or 10,000s of devices)
- MEMS-devices w/ storage and compute, everything is "active"

Detail Slides

Amdahl's Law

$$serial = S$$

$$parallel = \frac{(1-p)\cdot S + \frac{p\cdot S}{n}}{S}$$

Speedup in a Parallel System

- *p* is parallel fraction
- *(1 - p)* serial fraction is not improved

Database - Select

l_orderkey	l_shipdate	l_qty	l_price
7010	10-05-98	23	29356.3
36070	11-27-98	31	34167.9

```
select * from lineitem
where l_shipdate > '01-01-1998'
```

relation S

l_orderkey	l_shipdate	l_qty	l_price	l_disc
1730	01-25-93	6	11051.6	0.02
3713	04-12-96	32	29600.3	0.07
7010	10-05-98	23	29356.3	0.09
...				
32742	05-05-95	8	9281.9	0.01
36070	11-27-98	31	34167.9	0.04

Bloom Join

Use only Bloom filter at disks

- **semi-join only, final join at host**
- **fixed-size bit vectors - memory size O(1)!**

Query	**Join**	**Size of Bloom filter**					**Keys**	**Table**
		128 bits	8 kilobytes	64 kilobytes	1 megabyte	ideal	MB	GB
Q3	1.1	1.00	0.33	0.33	0.33	0.21	12.4	4.2
Q5	4.1	0.90	0.22	0.22	0.22	0.22	0.9	0.3
Q9	1.1	1.00	0.11	0.11	0.11	0.05	4.0	4.7
Q10	2.1	-	0.33	0.21	0.21	0.08	21.9	28.6

Memory size required at each disk

- **from TPC-D queries at 100 GB scale factor**
- **using a single hash function for all tables and keys**

Outline

Motivation

Computation in Storage

Performance Model

Applications & Prototype

Software Structure

Drive-Specific Functionality

Related Work

Contributions & Future Work

Database Primitives

Scan

- **evaluate predicate, return matching records**
- **low memory requirement**

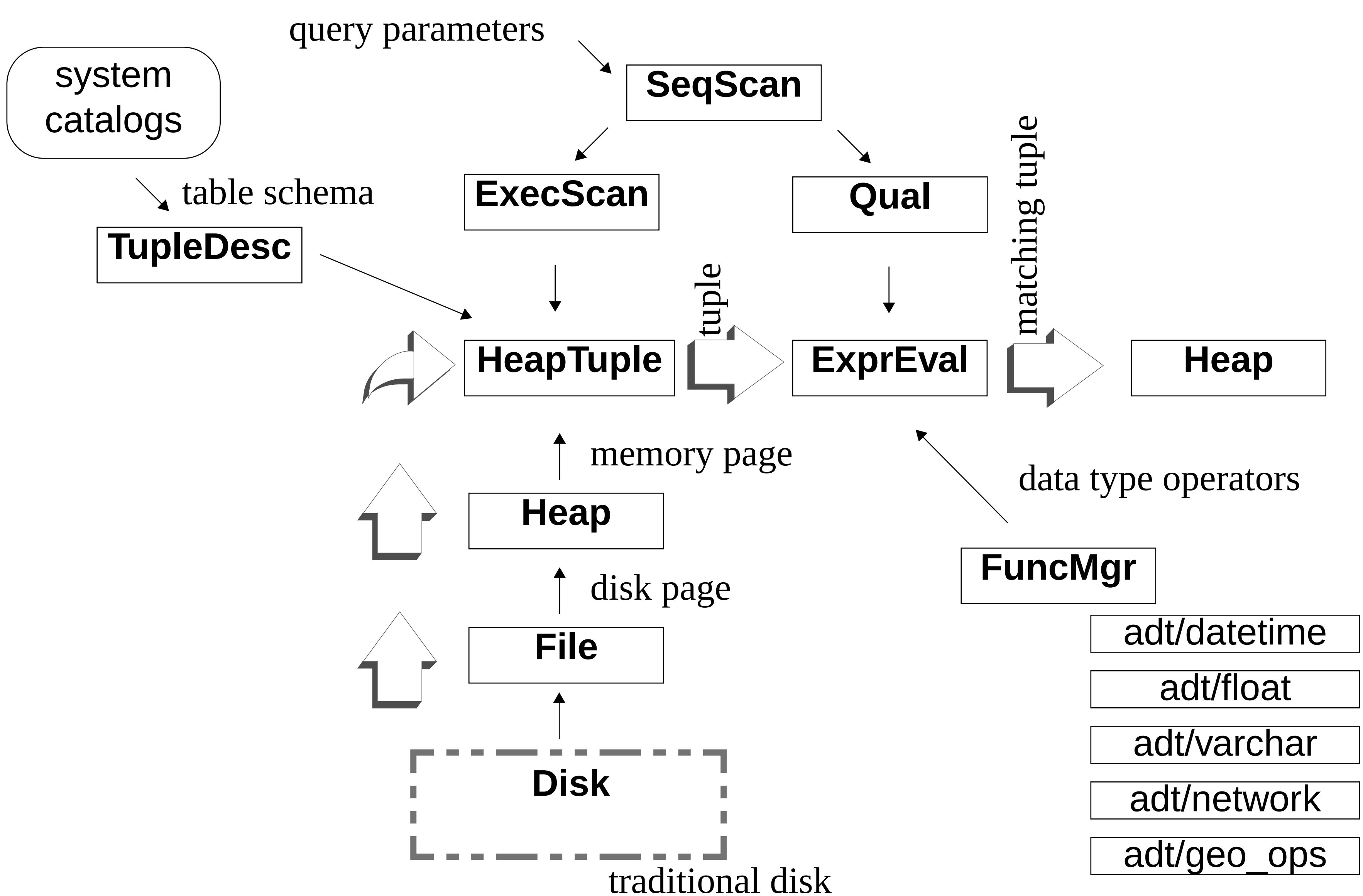
Join

- **identify matching records in semijoin**
- **via direct table lookup**
- **or Bloom filter, when memory is limited**

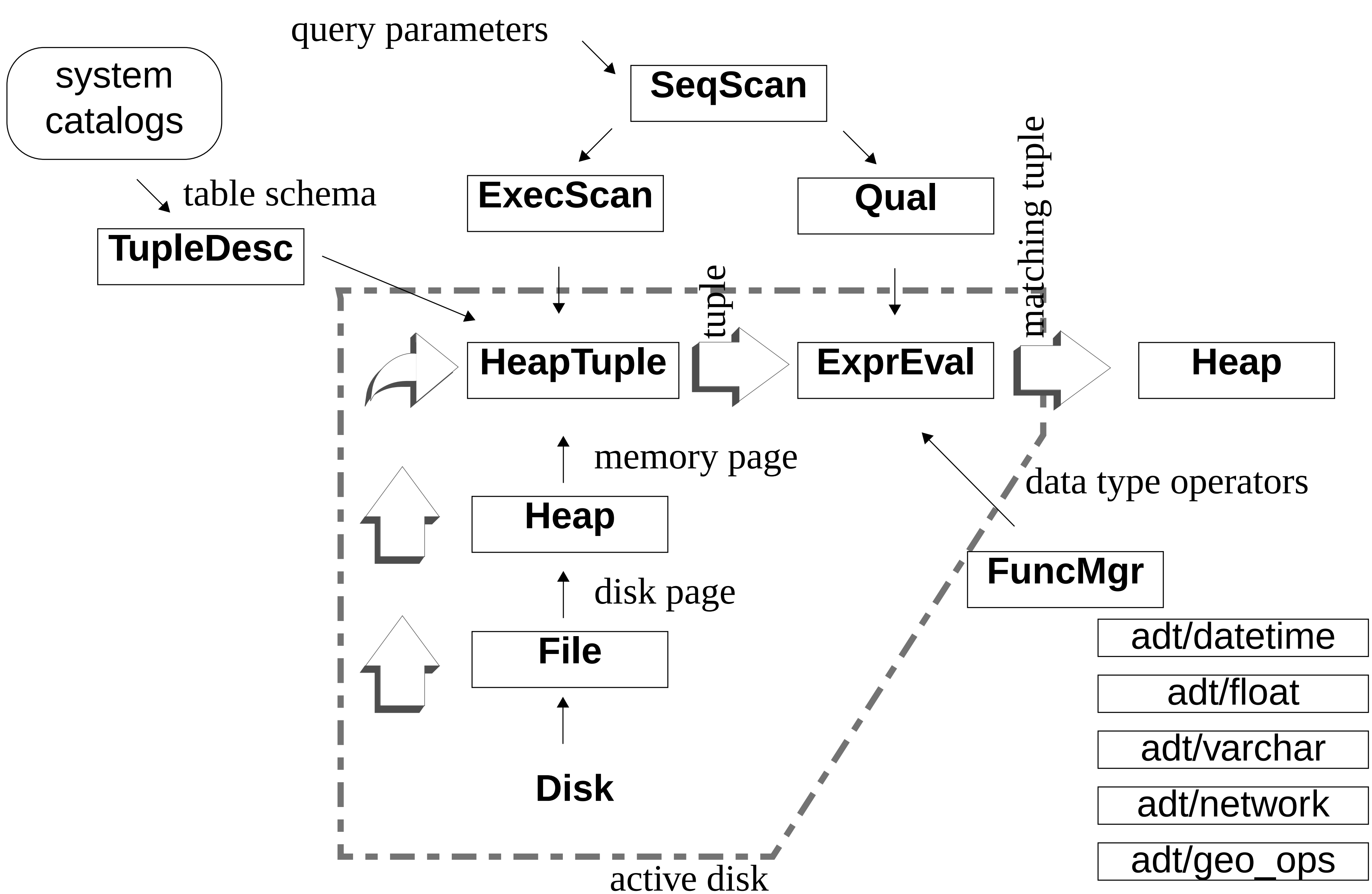
Aggregate/Sort

- **replacement selection with record merging**
- **memory size proportional to result, not input**
- **runs of length *2m* when used in full mergesort**

Execute Node

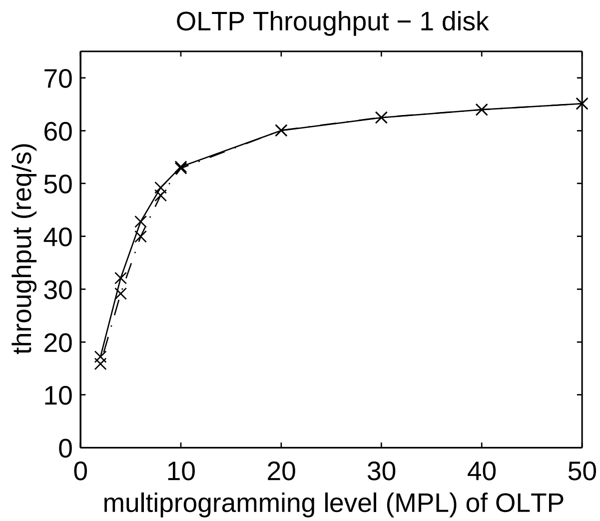


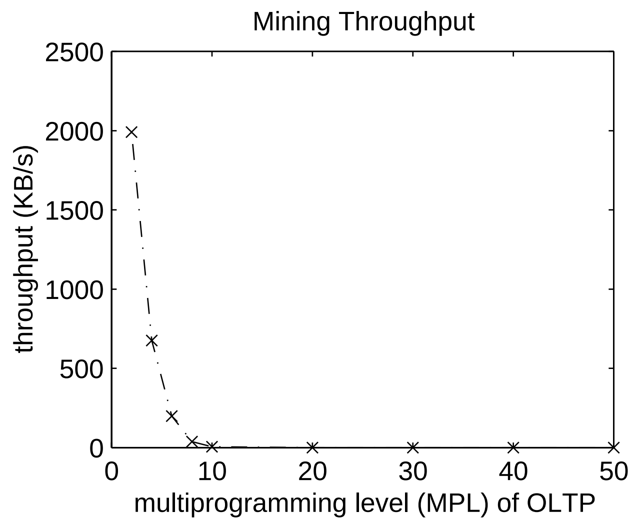
Active Disk Structure

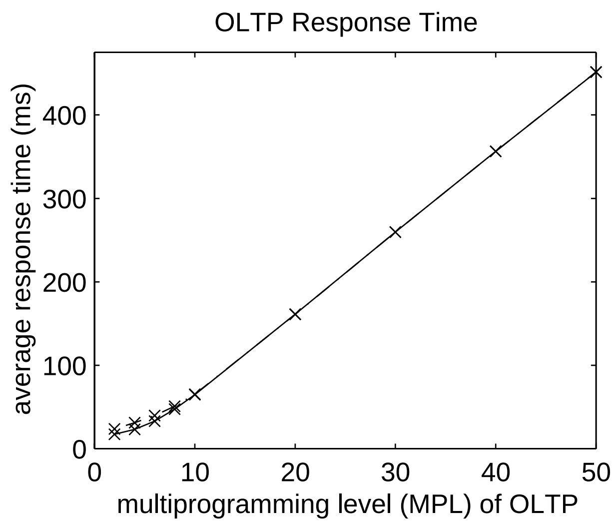


Data Mining for Free

- **read background blocks only when queue is empty**





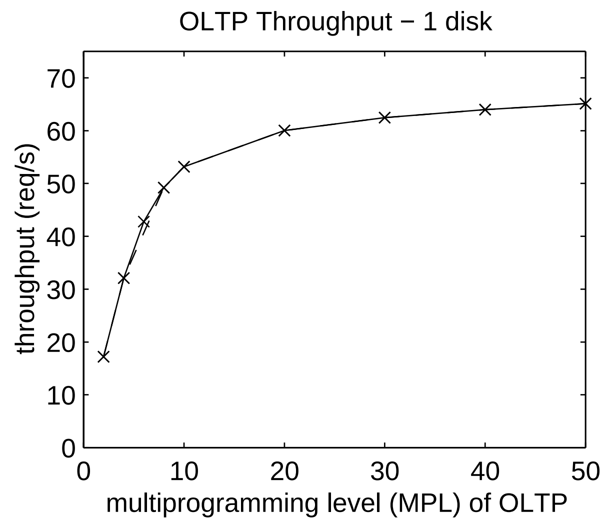


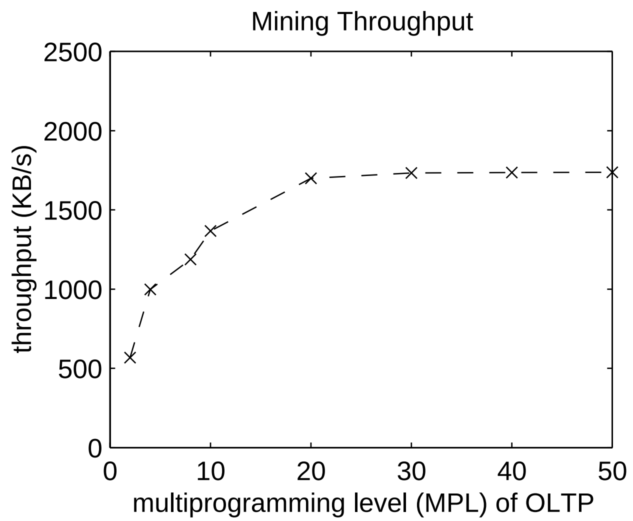
Background scheduling

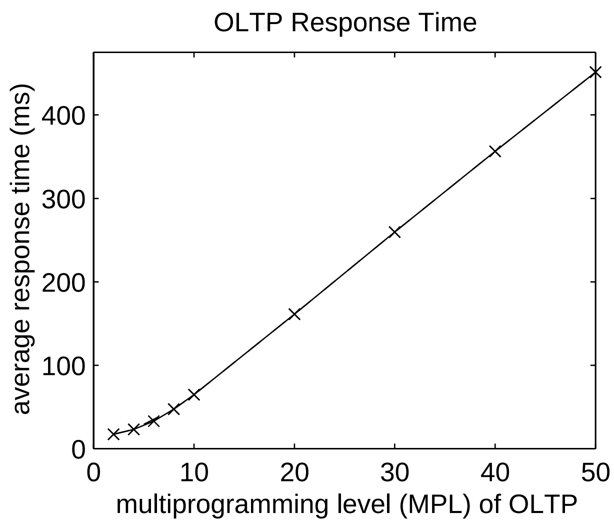
- **vary multiprogramming level - total number of pending requests**
- **background forced out at high foreground load**
- **up to 30% response time impact at low load**

Data Mining for Free

- **read background blocks only when completely "free"**





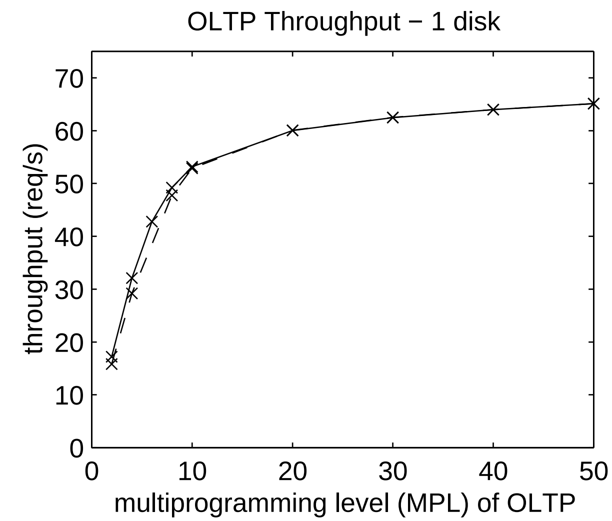


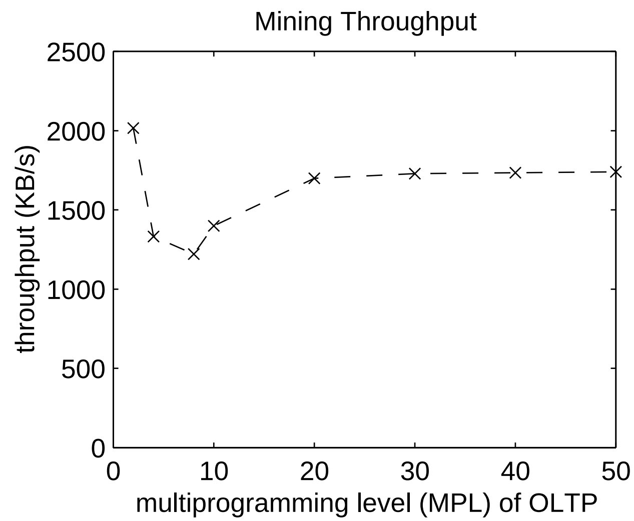
Free block scheduling

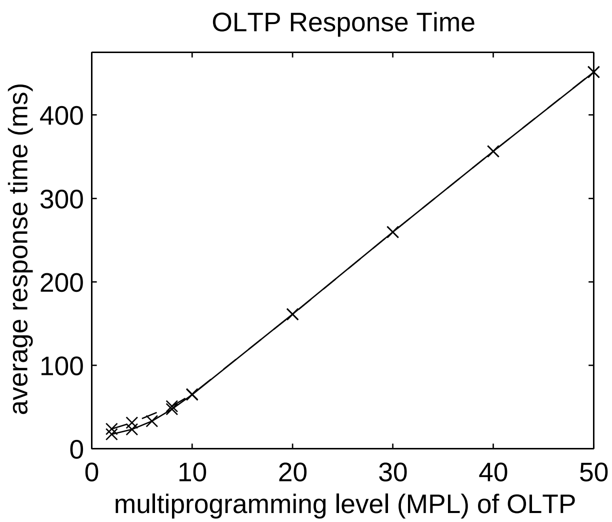
- **opportunistic read**
- **constant background bandwidth, even at highest loads**
- **no impact on foreground respond time**

Data Mining for Free

- **combine background and "free" blocks**







Integrated scheduling

- **possible only at drives**
- **combines application-level and disk-level information**
- **achieves 30% of the drives sequential band-width "for free"**

Extra Slides

Why Isn't This Parallel Programming?

It is

- parallel cores
- distributed computation
- serial portion needs to be small

Disks are different

- must protect the data, can't "just reboot"
- must continue to serve demand requests
- memory/CPU ratios driven by cost, reliability, volume
- come in boxes of ten
- *basic advantage* - compute close to the data

Opportunistically use this power

- e.g. data mining possible on an OLTP system
- ok to "waste" the power if it can't be used

Application Characteristics

Critical properties for Active Disk performance

- **cycles/byte => maximum throughput**
- **memory footprint**
- **selectivity => network bandwidth**

application	**input**	**computation** (instr/byte)	**throughput** (MB/s)	**memory** (KB)	**selectivity** (factor)	**bandwidth** (KB/s)
Select	m=1%	7	28.6	-	100	290
Search	k=10	7	28.6	72	80,500	0.4
Frequent Sets	s=0.25%	16	12.5	620	15,000	0.8
Edge Detection	t=75	303	0.67	1776	110	6.1
Image Registration	-	4740*	0.04	672	180	0.2
Select	m=20%	7	28.6	-	5	5,700
Frequent Sets	s=0.025%	16	12.5	2,000	14,000	0.9
Edge Detection	t=20	394	0.51	1750	3	170

Sorts

Local Sort Phase

- **replacement selection in Active Disk memory**

  process as data comes off the disk

- **build sorted runs of average size *2m***
- **can easily adapt to changes in available memory**

Local Merge Phase

- **perform sub-merges at disks**

  less runs to process at host

- **also adaptable to changes in memory**

Global Merge Phase

- **moves all data to the host and back**

Optimizations

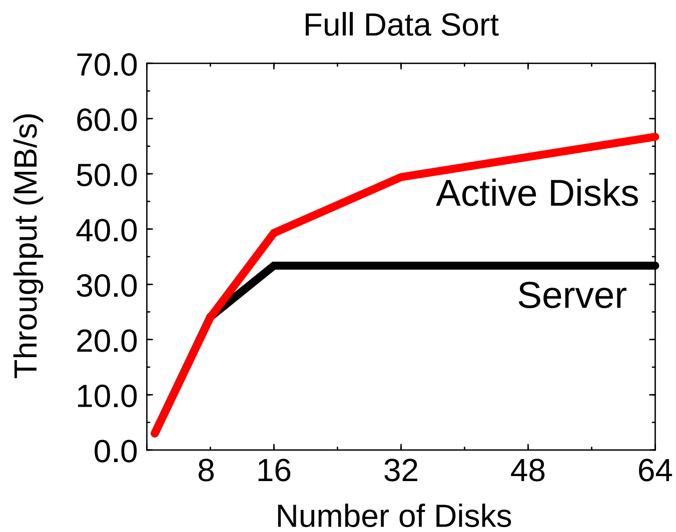
- **duplicate removal, aggregation lower requirements**

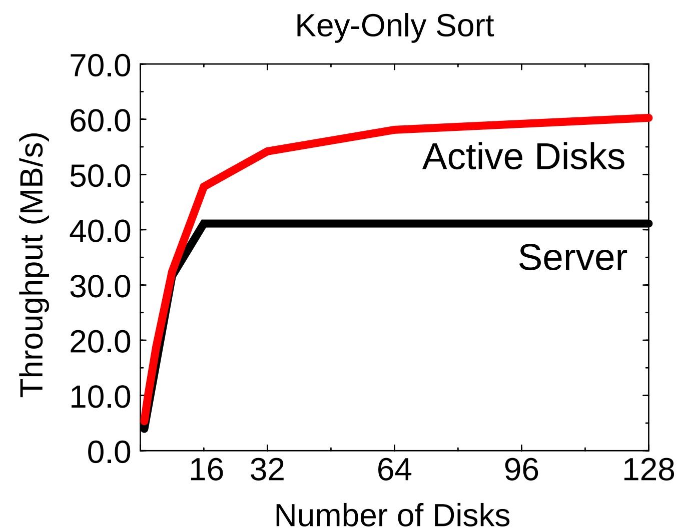
  memory required only for result, not source relations

Bottleneck is the Network - the Data *Must* Move Once

- **so goal is optimal utilization of links**

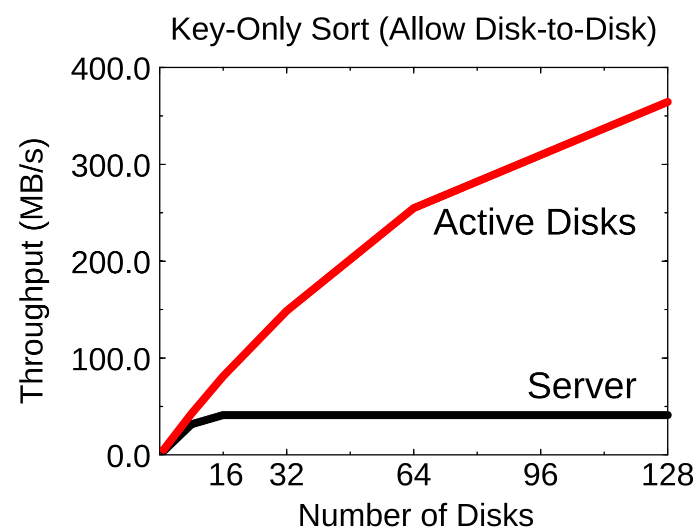
Sort Performance





Network is the bottleneck

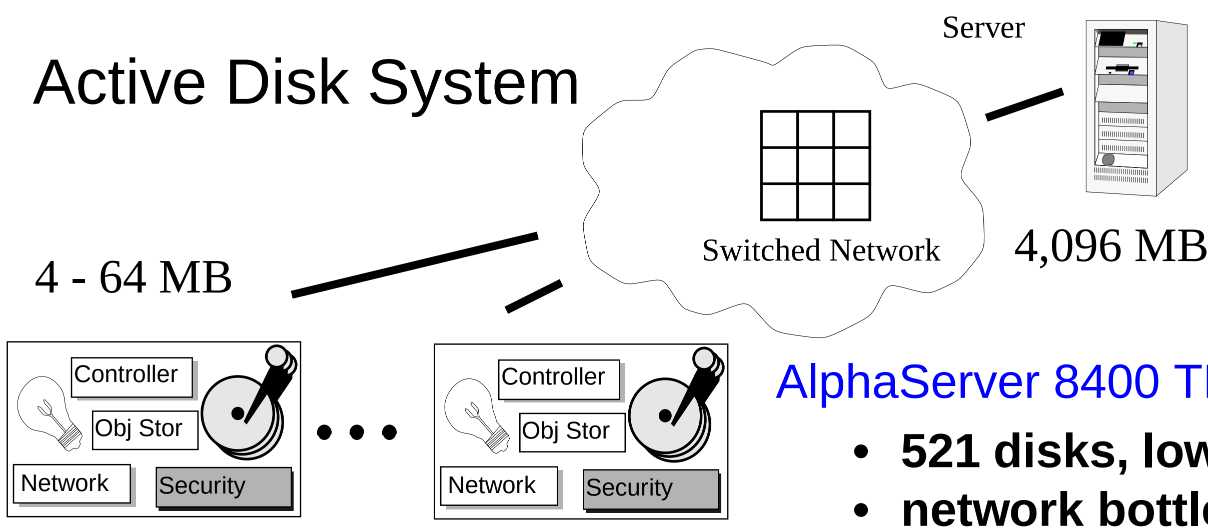
- **Active Disks benefit from reduced interconnect traffic**
- **using key-only sort improves both systems**
- **with direct disk to disk transfers, data *never* goes to the host**



Database - Joins

Size of R determines Active Disk partitioning

- **if |R| << |S| (R is the inner, smaller relation)**
  - **- and |R| < |*Active Disk* memory|**
    - **embarassingly parallel, linear speedup**
  - **- and |R| < |*Server* memory|**
    - **retain portion of R at each disk, and “assist” Server**
- **if |R| ~ |S| and |R| > |*Server* memory|**
  - **process R in parallel, minimize network traffic**
- **pre-join scan on S and R is always a win**
  - **reduces interconnect traffic**



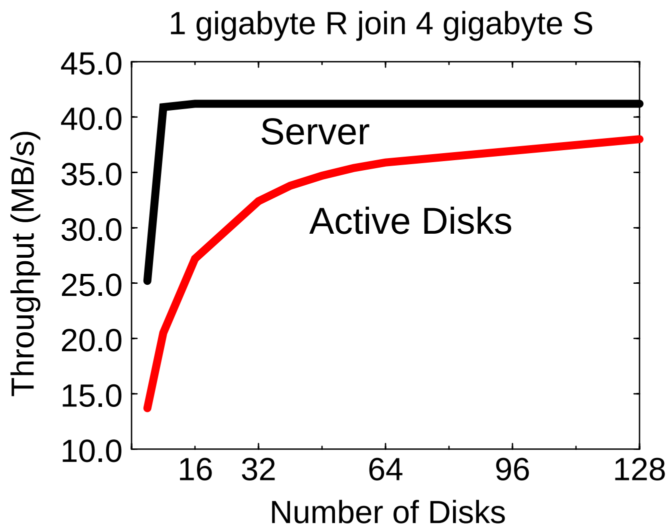
Assumptions

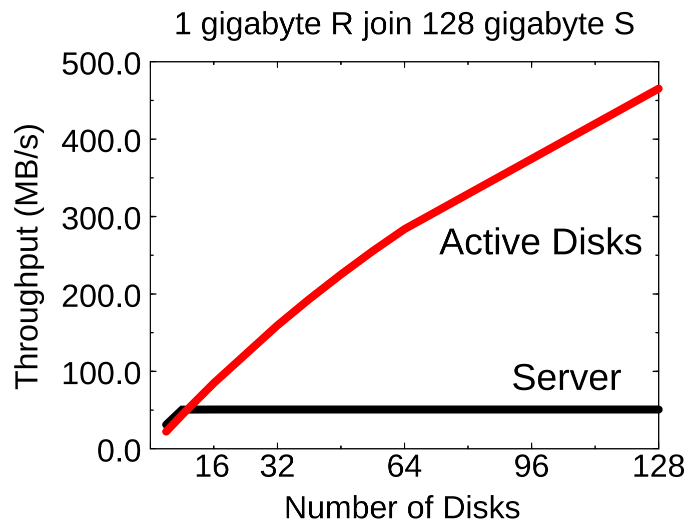
- **non-indexed keys**
- **not partition keys**
- **large S (multi-GB)**
  =>   many disks

AlphaServer 8400 TPC-D

- **521 disks, low CPU cost**
- **network bottlenecked**

Join Performance





benefits from reduced interconnect traffic

- determinant is relative size of inner and outer relations
- savings in network transfer
- vs. multiple passes at disks

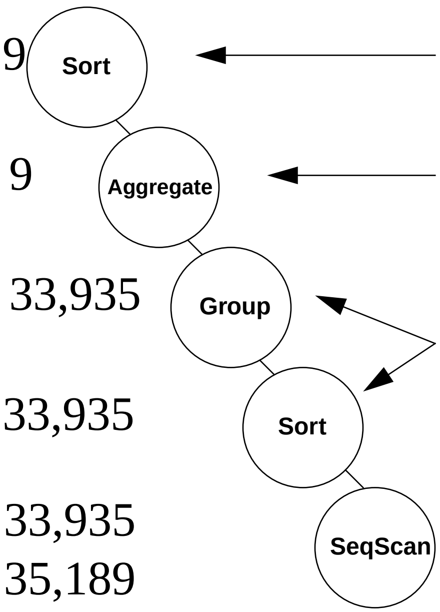
Database - TPC-D Query 1

Data Reduction

Sort	9 -> 9
Aggr	33,935 -> 9
Group	33,935 -> 33,935
Sort	33,935 -> 33,935
Qual	35,189 -> 33,935
Scan	126,440 -> 35,189

126,440 KB (15,805 pages) on disk

Query Plan



Query Text

```
order by l_returnflag, l_linestatus

sum(l_quantity), sum(l_price),
sum(l_price*(1-l_disc)),
sum(l_price*(1-l_disc)*(1+l_tax)),
avg(l_quantity), avg(l_price),
avg(l_disc), count(*)

group by l_returnflag, l_linestatus

where l_shipdate <= '1998-09-02'

select l_returnflag, l_linestatus,

from lineitem
```

input
output

Query Result

```
l_rf|l_ls|sum_qty|sum_base_price|sum_disc_price|    sum_charge|avg_qty|   price| disc| count
----+----+-------+--------------+--------------+--------------+-------+--------+-----+------
A   |F   |3773034| 5319329289.67| 5053976845.78| 5256336547.67| 25.509|35964.01|0.049|147907
N   |F   | 100245|  141459686.10|  134380852.77|  139710306.87| 25.625|36160.45|0.050|  3912
N   |O   |7464940|10518546073.97| 9992072944.46|10392414192.06| 25.541|35990.12|0.050|292262
R   |F   |3779140| 5328886172.98| 5062370635.93| 5265431221.82| 25.548|36025.46|0.050|147920
(4 rows)
```

Database - Data Reduction

Data Reduction for Sequential Scan and Aggregation

Query	Input Data (KB)	SeqScan Result (KB)	SeqScan Savings (selectivity)	Aggregate Result (bytes)	Aggregate Savings (selectivity)
Q1	126,440	34,687	3.6	240	147,997.9
Q4	29,272	86	340.4	80	1100.8
Q6	126,440	177	714.4	8	22,656.0

Input Table

```
l_okey|l_quantity| l_price|l_disc|l_tax|l_rf|l_ls|l_shipdate|l_commitdate|l_receiptdate|l_shipmode|l_comment
------+----------+--------+------+-----+----+----+----------+------------+-------------+----------+---------
  1730|         6|11051.58|  0.02|    0|N   |O   |09-02-1998|  10-10-1998|   09-13-1998|TRUCK     |wSRnnCx2
  3713|        32|29600.32|  0.07| 0.03|N   |O   |09-02-1998|  06-11-1998|   09-28-1998|TRUCK     |MOgnCO1
  7010|        23|29356.28|  0.09| 0.06|N   |O   |09-02-1998|  08-01-1998|   09-14-1998|MAIL      |jPNQlx3i
 19876|         4| 6867.24|  0.09| 0.08|N   |O   |09-02-1998|  09-06-1998|   09-29-1998|AIR       |3nRkNn4
 24839|         8|12845.52|  0.05| 0.02|N   |O   |09-02-1998|  10-14-1998|   09-06-1998|REG AIR   |jlw61g3
 25217|        10| 18289.1|  0.05| 0.07|N   |O   |09-02-1998|  08-12-1998|   09-26-1998|TRUCK     |SQ7xS5
 29348|        29|41688.08|  0.05| 0.02|N   |O   |09-02-1998|  07-04-1998|   09-18-1998|FOB       |C0NxhzM
 32742|         8| 9281.92|  0.01| 0.03|N   |O   |09-02-1998|  07-17-1998|   09-19-1998|FOB       |N3MO1C
 36070|        31|34167.89|  0.04|    0|N   |O   |09-02-1998|  07-11-1998|   09-21-1998|REG AIR   |k10wyR
[...more...]
(600752 rows)
```

Database - Aggregation

Data Reduction

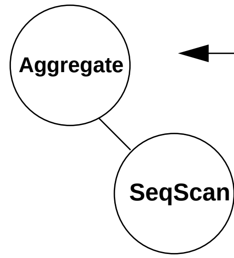
Aggr 43 -> 1

Qual   9,383 -> 43

Scan 126,440 -> 9,383

126,440 KB (15,805 pages) on disk

Query Plan



Query Text

```
select

sum(l_price*l_disc)

where l_shipdate >= '1994-01-01'
and l_shipdate < '1995-01-01'
and l_disc between 0.05 and 0.07
and l_quantity < 24

from lineitem
```

input
output

Query Result

```
revenue
-----------
11450588.04
(1 row)
```

Database - Partitioning

How to split operations between host and drives?

Answer: Use existing query optimizer

- operation costs
- per-table and per-attribute statistics
- ok if they are slightly out-of-date, only an estimate

Query	Input Data (KB)	Scan Result (KB)	Optimizer Estimate (KB)	Qualifier Result (KB)	Optimizer Estimate (KB)	Aggregate Result (bytes)	Optimizer Estimate (bytes)
Q1	126,440	35,189	35,189	34,687	33,935	240	9,180
Q4	29,272	2,343	2,343	86	141	80	64
Q6	126,440	9,383	9,383	177	43	8	8

Move ops to drives if there are sufficient resources

- if selectivity and parallelism overcome slower CPU

Be prepared to revert to host as two-stage algorithm

- consider the disk as “pre-filtering”
- still offloads significant host CPU and interconnect

Database - Optimizer Statistics

Statistics

starelid	staattnum	staop	stalokey	stahikey
18663	1	66	1	600000
18663	2	66	1	20000
18663	3	66	1	1000
18663	4	66	1	7
18663	5	295	1	50
18663	6	295	901	95949.5
18663	7	295	0	0.1
18663	8	295	0	0.08
18663	***9***	***1049***	***A***	***R***
18663	10	1049	F	O
18663	11	1087	01-02-1992	12-01-1998
18663	12	1087	01-31-1992	10-31-1998
18663	13	1087	01-08-1992	12-30-1998
18663	14	1049	COLLECT COD	TAKE BACK RETURN
18663	15	1049	AIR	TRUCK
18663	16	1049	0B6wmAww2Pg	zzzyRPS40ABMRSzmPyCNzA6

[...more...]
(61 rows)

estimate 17 output tuples

Attributes

estimate 4 output tuples

attrelid	attname	atttypid	attdisbursion	attlen	attnum
18663	l_orderkey	23	2.33122e-06	4	1
18663	l_partkey	23	1.06588e-05	4	2
18663	l_suppkey	23	0.000213367	4	3
18663	l_linenumber	23	0.0998572	4	4
18663	l_quantity	701	0.00434997	8	5
18663	l_extendedprice	701	2.66427e-06	8	6
18663	l_discount	701	0.0247805	8	7
18663	l_tax	701	0.0321099	8	8
18663	***l_returnflag***	***1042***	***0.307469***	***-1***	***9***
18663	l_linestatus	1042	0.300911	-1	10
18663	l_shipdate	1082	8.94076e-05	4	11
18663	l_commitdate	1082	8.33926e-05	4	12
18663	l_receiptdate	1082	8.90733e-05	4	13
18663	l_shipinstruct	1042	0.100238	-1	14
18663	l_shipmode	1042	0.0451101	-1	15
18663	l_comment	1042	0	-1	16

[...more...]
(572 rows)

Active PostgreSQL - Code Changes

Module	Original		Modified Host (New & Changed)		Active Disk	
	Files	Code	Files	Code	Files	Code
access	72	26,385	-	-	1	838
bootstrap	2	1,259	-	-	-	-
catalog	43	13,584	-	-	-	-
commands	34	11,635	-	-	-	-
executor	49	17,401	9	938	4	3,574
parser	31	9,477	-	-	-	-
lib	35	7,794	-	-	-	-
nodes	24	13,092	-	-	6	4,130
optimizer	72	19,187	-	-	-	-
port	5	514	-	-	-	-
regex	12	4,665	-	-	-	-
rewrite	13	5,462	-	-	-	-
storage	50	17,088	1	273	-	-
tcop	11	4,054	-	-	-	-
utils/adt	40	31,526	-	-	2	315
utils/fmgr	4	2,417	-	-	1	281
utils	81	19,908	-	-	1	47
Total	578	205,448	10	1,211	15	9,185
					New	1,257

Code Specialization

query	**type**	**computation** (instr/byte)	**throughput** (MB/s)	**memory** (KB)	**selectivity** (factor)	**instructions** (KB)
Q1	aggregation	1.82	73.1	488	816	9.1/4.7
Q13	hash-join	0.15	886.7	576	967,000	14.3/10.5

Optimized Implementation

- **direct C code, single query only, raw binary files**
- **133 MHz Alpha 3000/400, Digital UNIX 3.2**

operation	**computation** (cycles/byte)	**throughput** (MB/s)	**selectivity** (factor)
Scan	28	17.8	4.00
Qualification	29	17.2	1.05
Sort/Group	71	7.0	1.00
Sort/Aggregate	196	2.5	3,770.00

Database System

- **database manager database is PostgreSQL 6.4.2**
- **much higher cycles/byte than direct C implementation**
  - **parses general SQL statements**
  - **handles arbitrary tuple formats**

History - SCAFS

SCAFS (Son of Content-Addressable File Store)

- processing unit in a 3.5” form factor, fit into a drive shelf
- communication via SCSI commands

Goals

- invisible to the application layer (i.e. hidden under SQL)
- established as an industry-standard for high volume market

Benefits

- 40% to 3x throughput improvement in a mixed workload
- 20% to 20x improvement in response time
- 2x to 20x for a “pure” decision support workload
- up to 100x improvement in response time

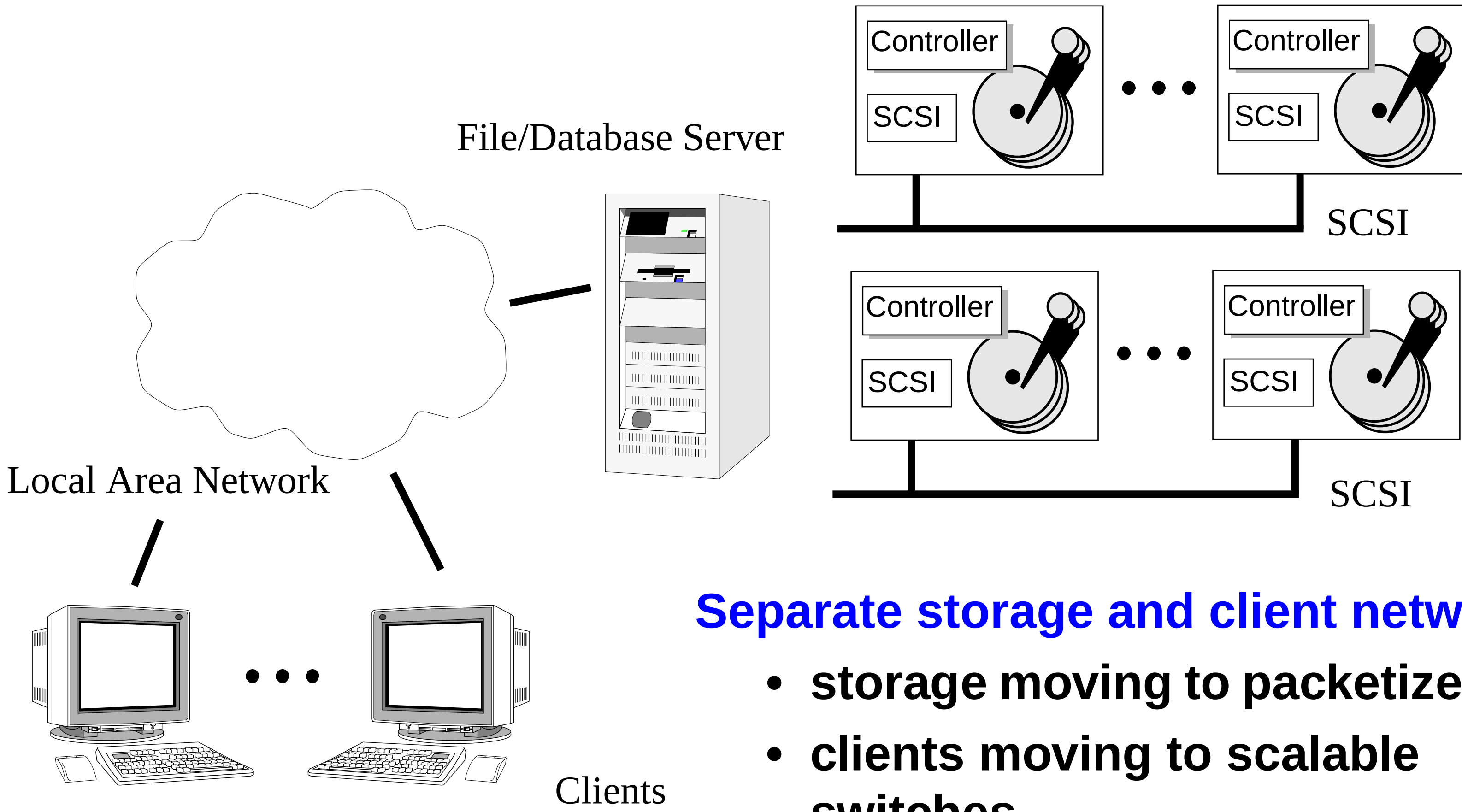
Lessons from CAFS [Anderson98]

Why did CAFS not become wildly popular?

- **"synchronization was a big problem"**
  Answer **- Yes. Major concern for OLTP, less for "mining".**
- **"dynamic switching between applications is a problem"**
  Answer **- Yes. But operating systems know how to do this.**
- **"not the most economical way to add CPU power"**
  Answer **- but it *is* the best bandwidth/capacity/compute combo and you can still add CPU if that helps (and if you can keep it fed)**
- **"CPU is a more flexible resource", disk processor wasted when not in use**
  Answer **- you're already wasting it today, silicon is everywhere**
- **"memory size is actually a bigger problem"**
  Answer **- use adaptive algorithms, apps have "sweet spots"**
- **"needed higher volume, lower cost function"**
  Answer **- this is exactly what the drive vendors can provide no specialized, database-specific hardware necessary**
- **"could not get it to fit into the database world"**
  Answer **- proof of concept, community willing to listen**

Yesterday's Server-Attached Disks

Store-and-forward data copy through server machine



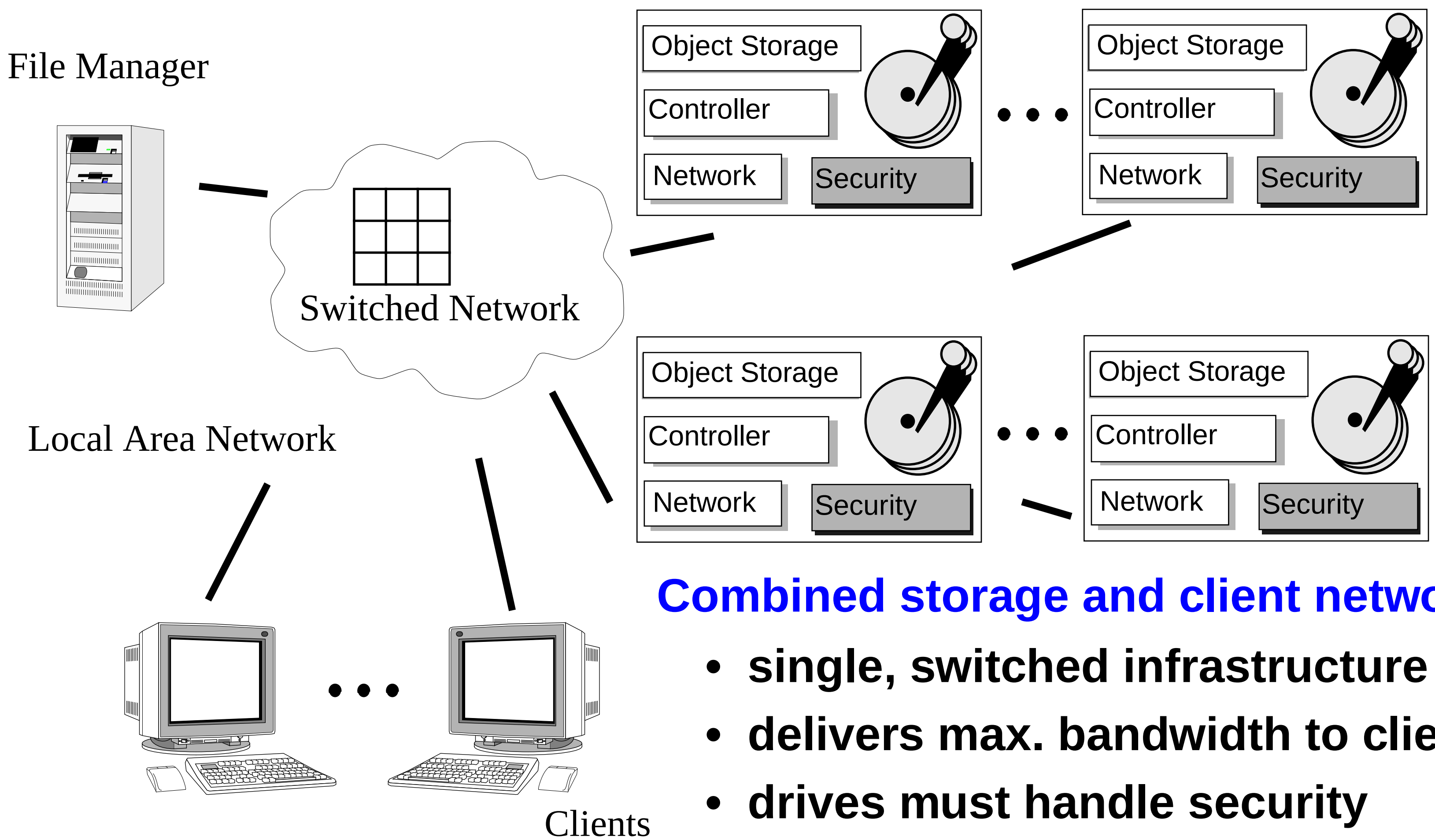
Separate storage and client networks

- **storage moving to packetized FC**
- **clients moving to scalable switches**

Network-Attached Secure Disks

Eliminate server bottleneck w/ network-attached

- **server scaling [SIGMETRICS '97]**
- **object interface, filesystems [CMU-CS '98]**
- **cost-effective, high bandwidth [ASPLOS '98]**



Combined storage and client networks

- **single, switched infrastructure**
- **delivers max. bandwidth to clients**
- **drives must handle security**

TPC-D Benchmark

Consists of *high selectivity*, *ad-hoc* queries

	entire query			scan only	
query	**input** (MB)	**result** (KB)	**selectivity** (factor)	**input** (MB)	**selectivity** (factor)
Q1	672	0.2	4.8 million	672	3.3
Q5	857	0.09	9.7 million	672	3.5
Q7	857	0.02	3.5 million	672	4.0
Q9	976	6.5	154,000	672	2.2
Q11	117	0.3	453,000	115	7.2

Scale Factor = 1 GB

Simple filtering on input

- **factors of 3x and more savings in load on interconnect**

Entire queries (including aggregation and joins)

- **factors of 100,000 and higher savings**

Implementation Issues

Partitioning

- combining disk code with “traditional” code

Mobility

- code must run on disks and/or host
- Java (!) (?)
  - + popular, tools (coming soon), strong typing
  - - somewhat different emphasis what to optimize for
- more “static” extensions

Interfaces

- capability system of NASD as a base
- additional inquiry functions for scheduling
- additional power (via capabilities) for storage mgmt

Value-Added Storage

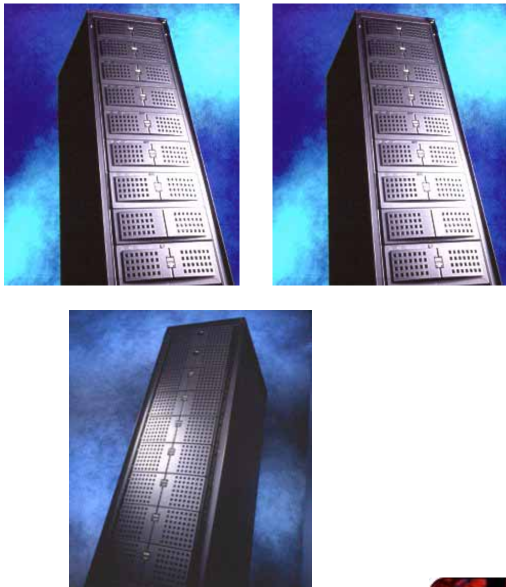
Variety of value-added storage devices

System	Function	Cost	Premium	Other
Seagate Cheetah 18LP LVD	disk only	$900	-	18 GB, lvd, 10,000 rpm
Seagate Cheetah 18LP FC	disk only	$942	5%	FC
Dell 200S PowerVault	drive shelves & cabinet	$10,645	48%	8 lvd disks
Dell 650F PowerVault	dual RAID controllers	$32,005	240%	10 disks, full FC
Dell 720N PowerVault	CIFS, NFS, Filer	$52,495	248%	16 disks, ether, 256/8 cache
EMC Symmetrix 3330-18	RAID, management	$160,000	962%	16 disks, 2 GB cache

Price premium

- cabinet cost is significant
- network-attached storage is as costly as RAID
- “management” gets the biggest margin

Network "Appliances" Can Win Today



Dell PowerEdge & PowerVault System

Dell PowerVault 650F $40,354 x 12 =484,248
512 MB cache, dual link controllers, additional 630F cabinet, 20 x 9 GB FC disks, software support, installation
Dell PowerEdge 6350 $11,512 x 12 = 138,144
500 MHz PIII, 512 MB RAM, 27 GB disk
3Com SuperStack II 3800 Switch 7,041
10/100 Ethernet, Layer 3, 24-port
Rack Space for all that 20,710

NASRaQ System

Cobalt NASRaQ $1,500 x 240 =360,000
250 MHz RISC, 32 MB RAM, 2 x 10 GB disks
Extra Memory (to 128 MB each) $183 x 360= 65,880
3Com SuperStack II 3800 Switch $7,041 x 11= 77,451
240/24 = 10 + 1 to connect those 10
Dell PowerEdge 6350 Front-End 11,512
Rack Space (estimate 4x as much as the Dells) 82,840
Installation & Misc 50,000

Comparison

	Dell	Cobalt
Storage	2.1 TB	4.7 TB
Spindles	240	480
Compute	6 GHz	60 GHz
Memory	12.3 GB	30.7 GB
Power	23,122 W	12,098 W
Cost	$650,143	$647,683