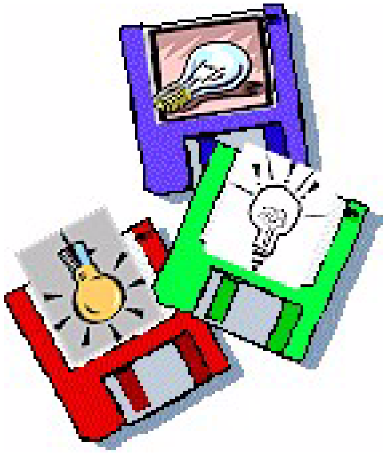
Active Disks for Databases



Erik Riedel

Parallel Data Laboratory,
Center for Automated Learning and Discovery
Carnegie Mellon University
www.pdl.cs.cmu.edu/Active

Outline

Opportunity & Background

Why Databases

Prototype - Performance

Prototype - Code Structure

History

Summary

Opportunity

Large database systems - lots of disks, lots of power

System	Processing (MHz)		Data Rate (MB/s)	
	CPU	Disks	I/O Bus	Disks
Compaq Proliant TPC-C	4 x 400=**1,600**	*141* x 200=**28,200**	**133**	**2,115**
Microsoft Terraserver	8 x 440=**3,520**	*324* x 200=**64,800**	**532**	**4,860**
Compaq AlphaServer 500 TPC-C	1 x 500=**500**	*61* x 200=**12,200**	**266**	**915**
Compaq AlphaServer 8400 TPC-D	12x612=**7,344**	*521* x 200=**104,200**	**532**	**7,815**

- **assume disk offers equivalent of 200 host MHz**
- **assume disk sustained data rate of 15 MB/s**

Lots more cycles and MB/s in disks than in host

- **main bottleneck is backplane I/O bandwidth**

Advantage - Active Disks

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

Basic advantages of an Active Disk system

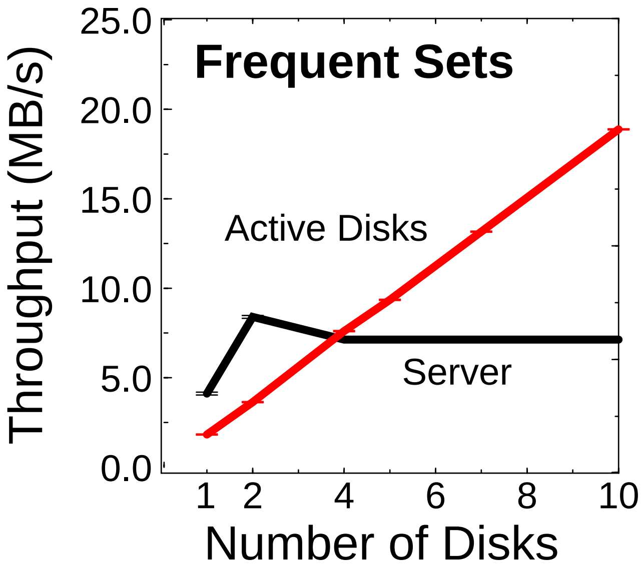
- **parallel processing** - lots of disks
- **bandwidth reduction** - filtering operations are common
- **scheduling** - little bit of computation can go a long way

Parameters for appropriate applications

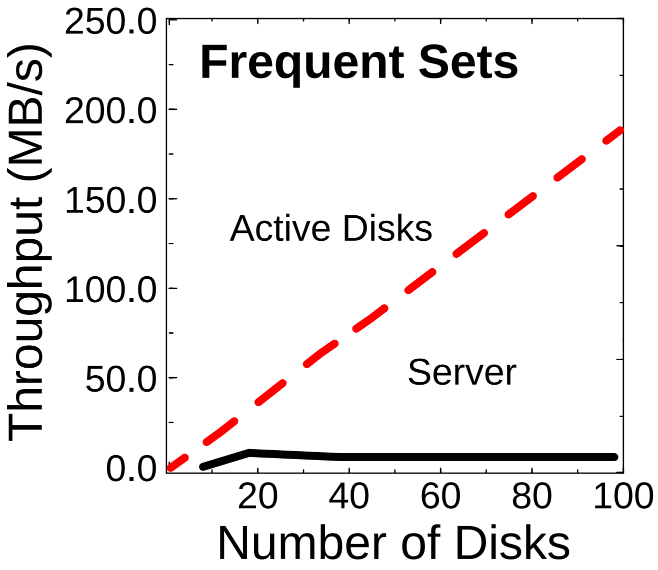
- execution time dominated by data-intensive “core”
- allows parallel implementation of “core”
- processing cycles per byte of data processed
- “selectivity” of processing
- memory footprint

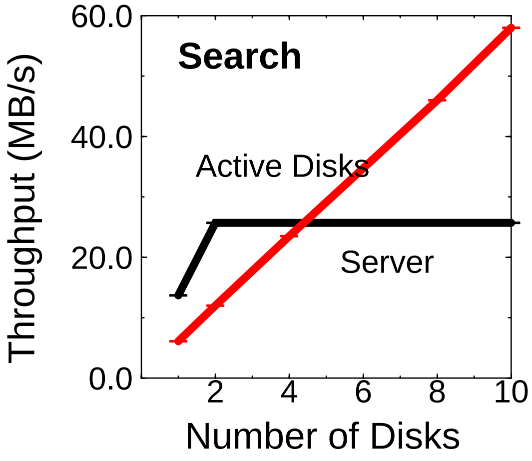
Data Mining & Multimedia [VLDB ‘98]

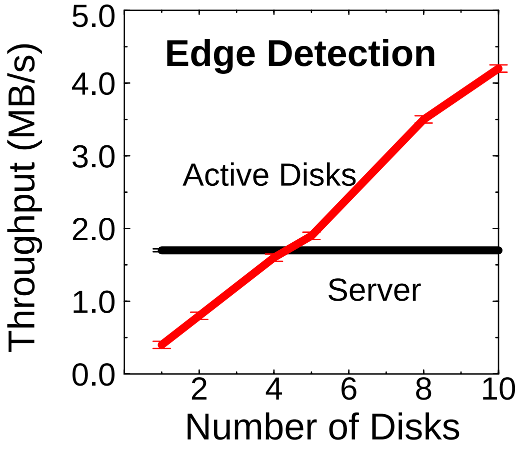
Prototype

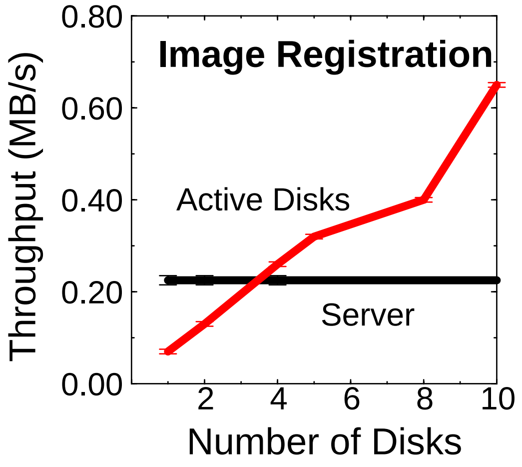


Scaling Up









Data Mining & Multimedia [VLDB '98]

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

Application Characteristics

Critical properties for Active Disk performance

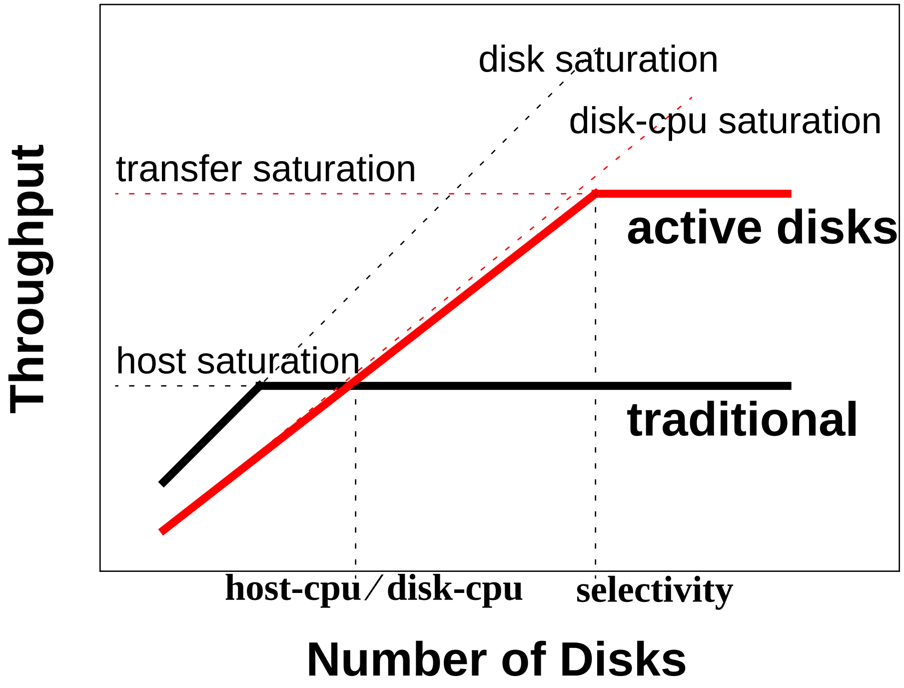
- cycles/byte => maximum throughput
- memory footprint
- selectivity => network bandwidth

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

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



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

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

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

Traditional server: $t = max\left(\frac{N_{\mathrm{in}}}{d \cdot r_d}, \frac{N_{\mathrm{in}}}{r_n}, \frac{N_{\mathrm{in}} \cdot w}{s_{\mathrm{cpu}}}\right)$ and throughput $= \frac{N_{\mathrm{in}}}{t} = min\left(d \cdot r_d, r_n, \frac{s_{\mathrm{cpu}}}{w}\right)$

Active Disks: $t_{active} = max\left(\frac{N_{\mathrm{in}}}{d \cdot r_d'}, \frac{N_{\mathrm{out}}}{r_n'}, \frac{N_{\mathrm{in}} \cdot w}{d \cdot s_{\mathrm{cpu}}'}\right)$ and throughput$_{\mathrm{active}}$ $= \frac{N_{\mathrm{in}}}{t_{active}} = min\left(d \cdot r_d', r_n' \cdot \frac{N_{\mathrm{in}}}{N_{\mathrm{out}}}, d \cdot \frac{s_{\mathrm{cpu}}'}{w}\right)$

Rewriting yields: throughput$_{\mathrm{active}}$ $= min\left(\alpha_d \cdot (d \cdot r_d), \alpha_N \cdot \alpha_n \cdot (r_n), d \cdot \alpha_s \cdot \left(\frac{s_{\mathrm{cpu}}}{w}\right)\right)$, Speedup:

For $1/\alpha_s < d < \alpha_N$, the speedup is:

$$S = \frac{d \cdot (s_{\mathrm{cpu}}'/w)}{min(r_n, s_{\mathrm{cpu}}/w)}$$

$$\geq d \cdot \alpha_s$$

and for $d > \alpha_N$, is:

$$S = \frac{(r_n' \cdot \alpha_N)}{min\left(r_n, \frac{s_{\mathrm{cpu}}}{w}\right)}$$

$$= max\left(\alpha_N \cdot \alpha_n, \alpha_N \cdot \alpha_s \cdot \left(\frac{w \cdot r_n'}{s_{\mathrm{cpu}}'}\right)\right)$$

$$> \alpha_N \cdot max(\alpha_n, \alpha_s)$$

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

Modified Performance Model

Traditional server:

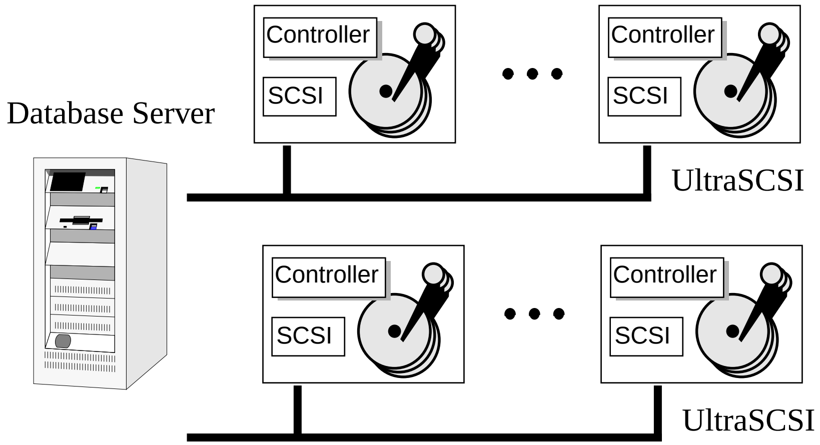
$$t = max\left(\frac{N_{\text{in}}}{d \cdot r_d}, \frac{N_{\text{in}}}{r_n}, \frac{N_{\text{in}} \cdot w}{s_{\text{cpu}}}\right) + serialfraction$$

Active Disks:

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

- **adds serial fraction**
- **fixed part of execution time**
- **not improved with additional disks**

Prototype Comparison

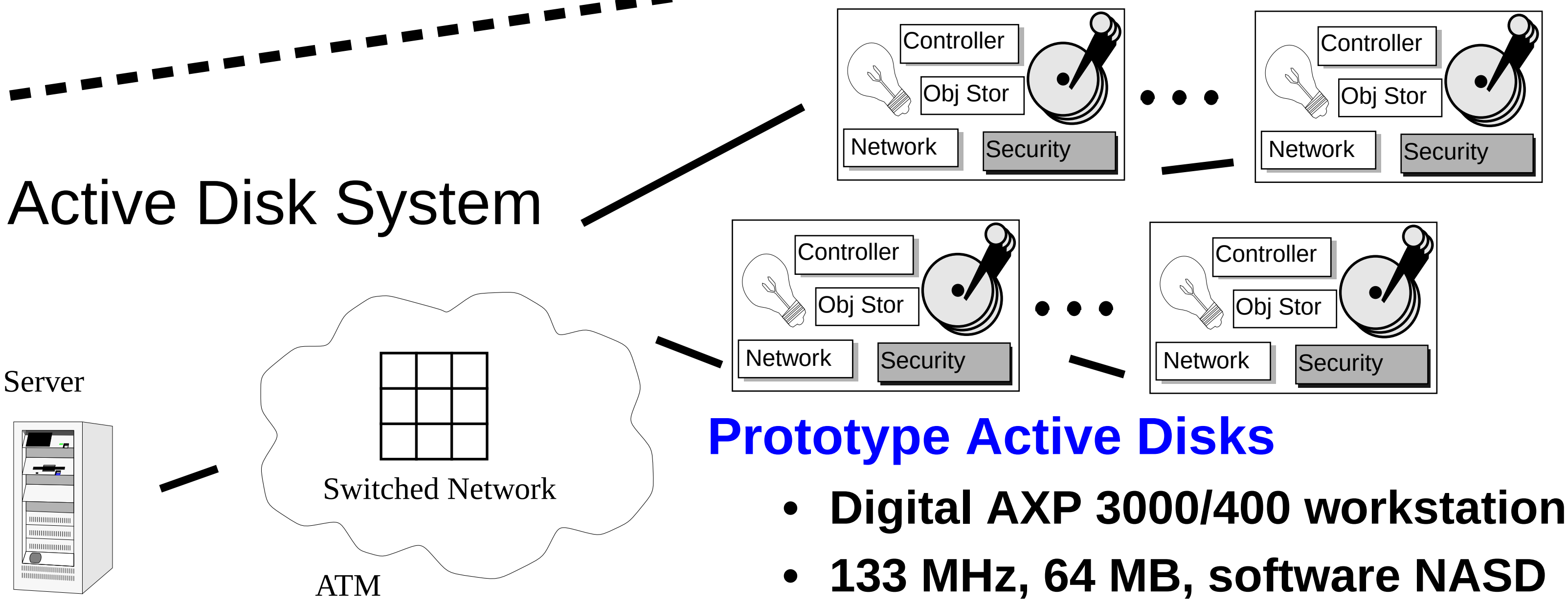


Traditional System

Digital AlphaServer 500/500

- **500 MHz, 256 MB memory**
- **disks - Seagate Cheetah**
- **4.5 GB, 10,000 RPM, 11.2 MB/s**

Active Disk System



Prototype Active Disks

- **Digital AXP 3000/400 workstation**
- **133 MHz, 64 MB, software NASD**
- **Seagate Medallist disks**

Objections to Active Disks for Database

“Performance benefits are too small”

- claim: parallelism just isn’t there

“Functionality is too complicated for Active Disks”

“Too difficult to change existing code”

“This has been tried before, and didn’t succeed then”

- database machines didn’t take over the world

“Can just do it with a bunch of PCs”

- cost argument, not covered here

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
- TPC-C transaction processing
  - not big data
  - operations per second
  - less dramatic speedups

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**

Objections to Active Disks for Database

“Performance benefits are too small”

- claim: parallelism just isn’t there

“Functionality is too complicated for Active Disks”

“Too difficult to change existing code”

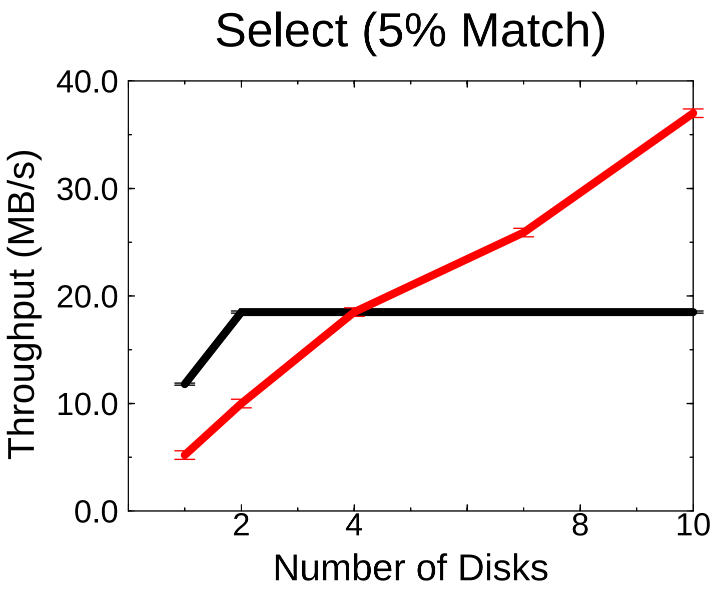
“This has been tried before, and didn’t succeed”

- database machines didn’t take over the world

“Can just do it with a bunch of PCs”

- cost argument, not covered here

Active PostgreSQL Select



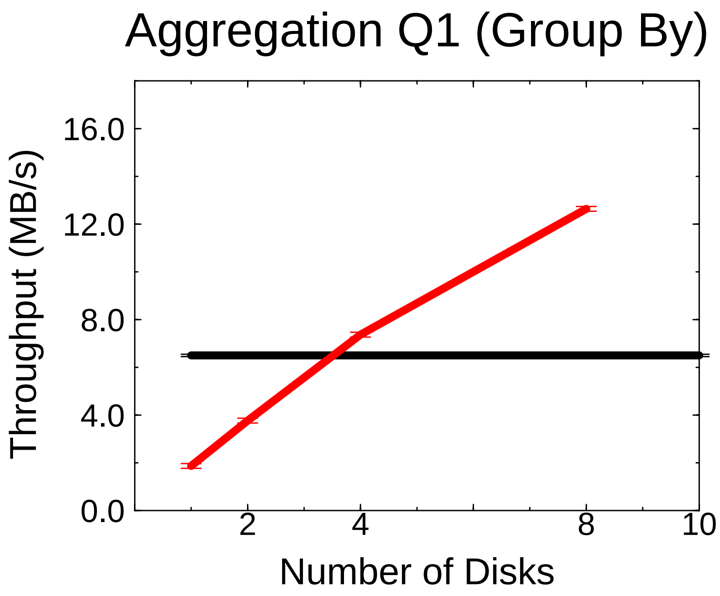
Experimental setup

- **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 Disks performance**

Active PostgreSQL Aggregation



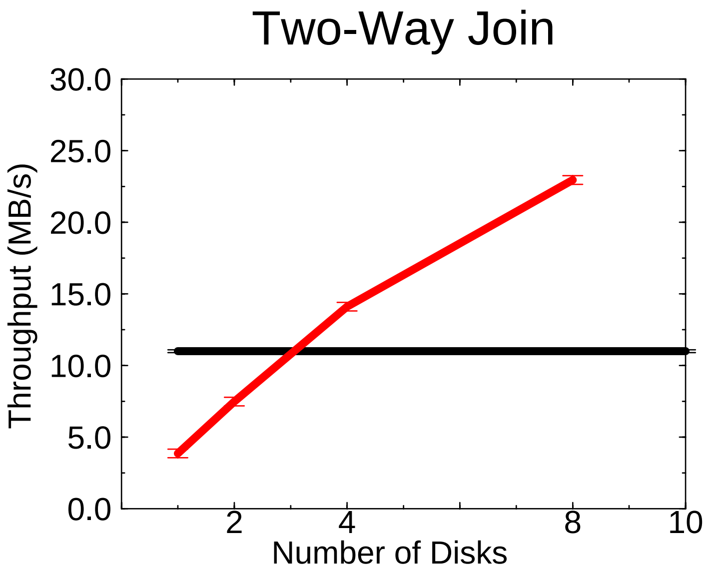
Experimental setup

- 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 `sum()...group by` operation (selectivity = 650)
- cycles/byte = 32
- crossover at four Active Disks (= 500 / 133)
- cpu limited

Active PostgreSQL Join



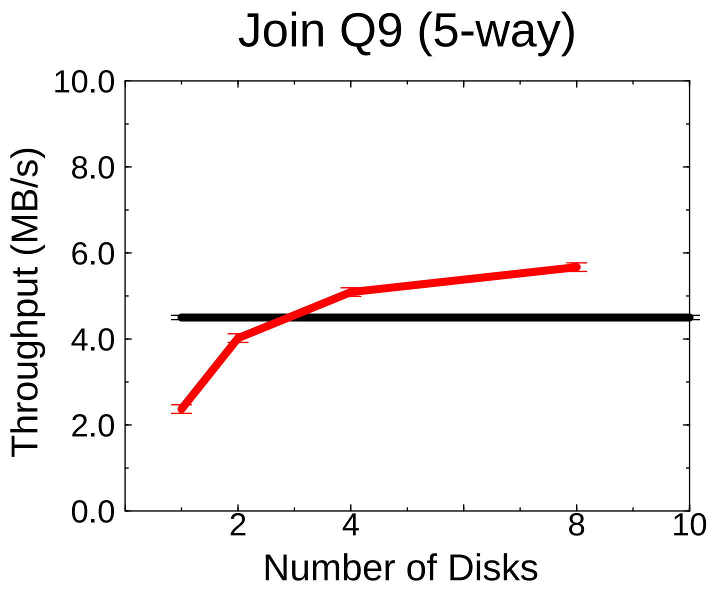
Experimental setup

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

Active PostgreSQL Join II



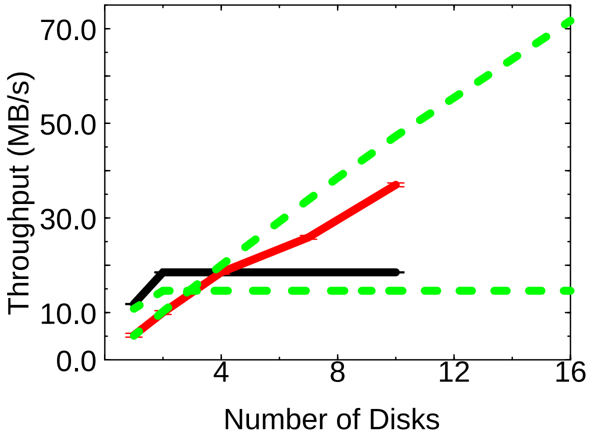
Experimental setup

- **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**

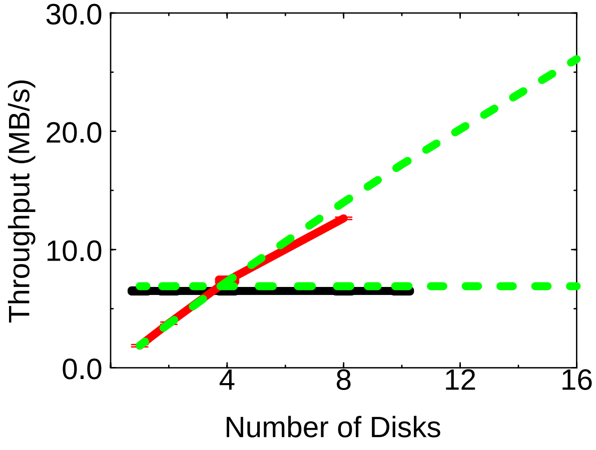
performance results

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

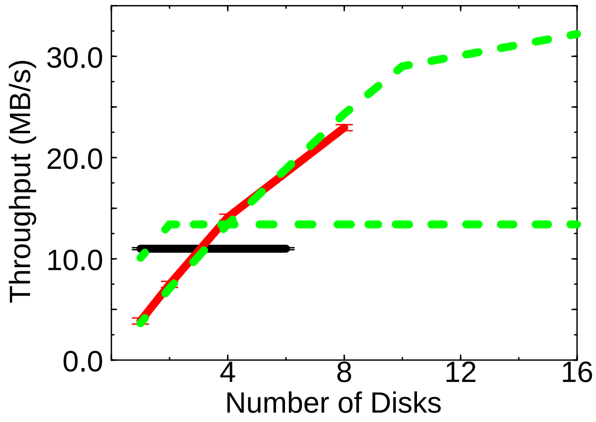
Model Validation (Database)



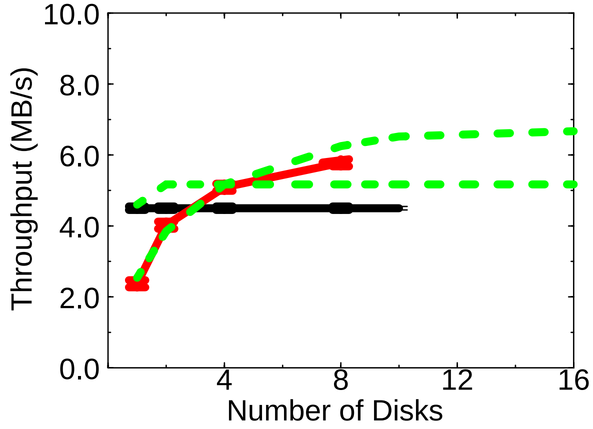
Select Q1 (5% Match)



Aggregation Q1 (Group By)



Two-Way Join



Join Q9

Objections to Active Disks for Database

“Performance benefits are too small”

- claim: parallelism just isn’t there

“Functionality is too complicated for Active Disks”

“Too difficult to change existing code”

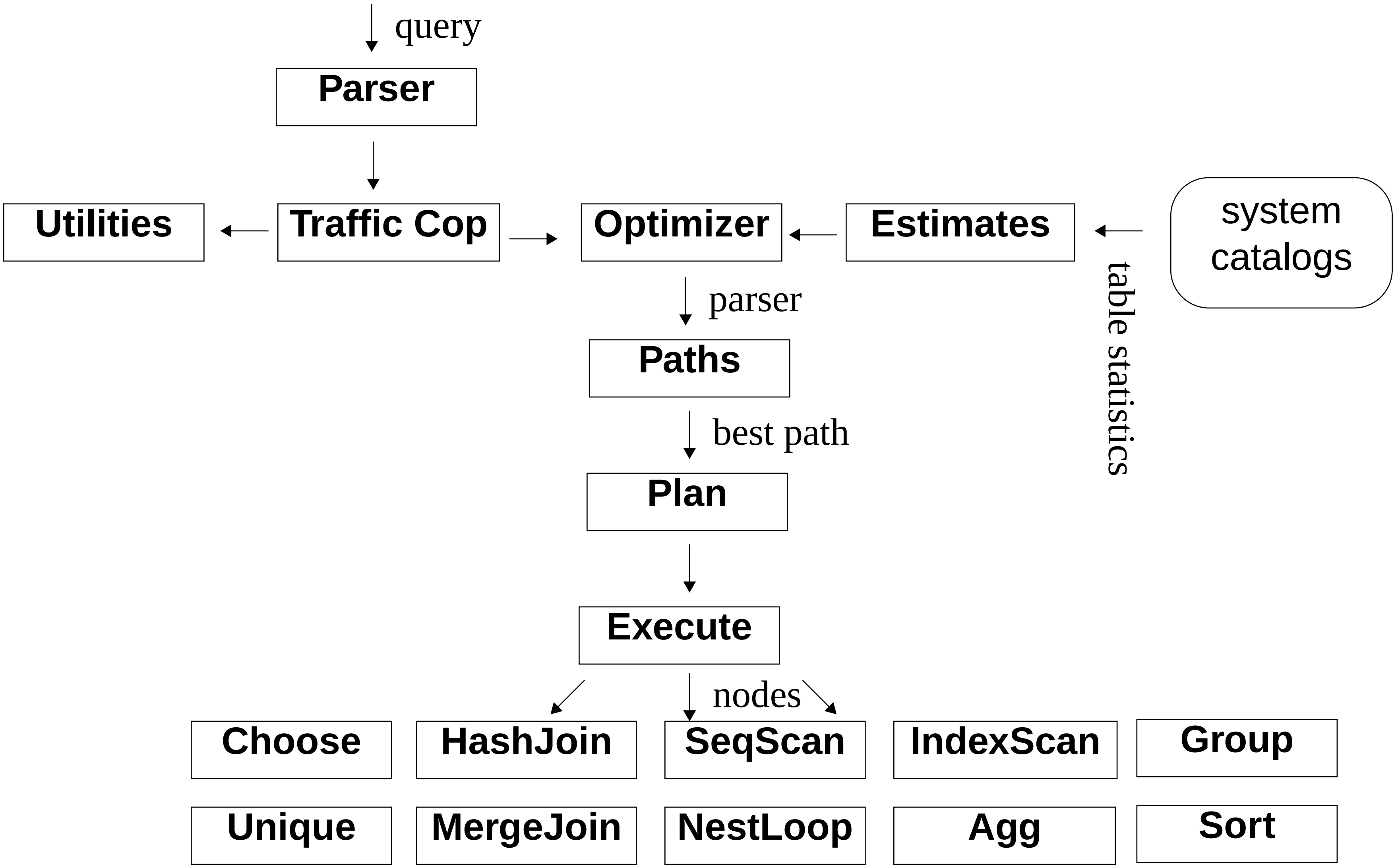
“This has been tried before, and didn’t succeed”

- database machines didn’t take over the world

“Can just do it with a bunch of PCs”

- cost argument, not covered here

PostgreSQL Software Structure



query

Parser

Utilities

Traffic Cop

Optimizer

Estimates

system catalogs

table statistics

parser

Paths

best path

Plan

Execute

nodes

Choose

HashJoin

SeqScan

IndexScan

Group

Unique

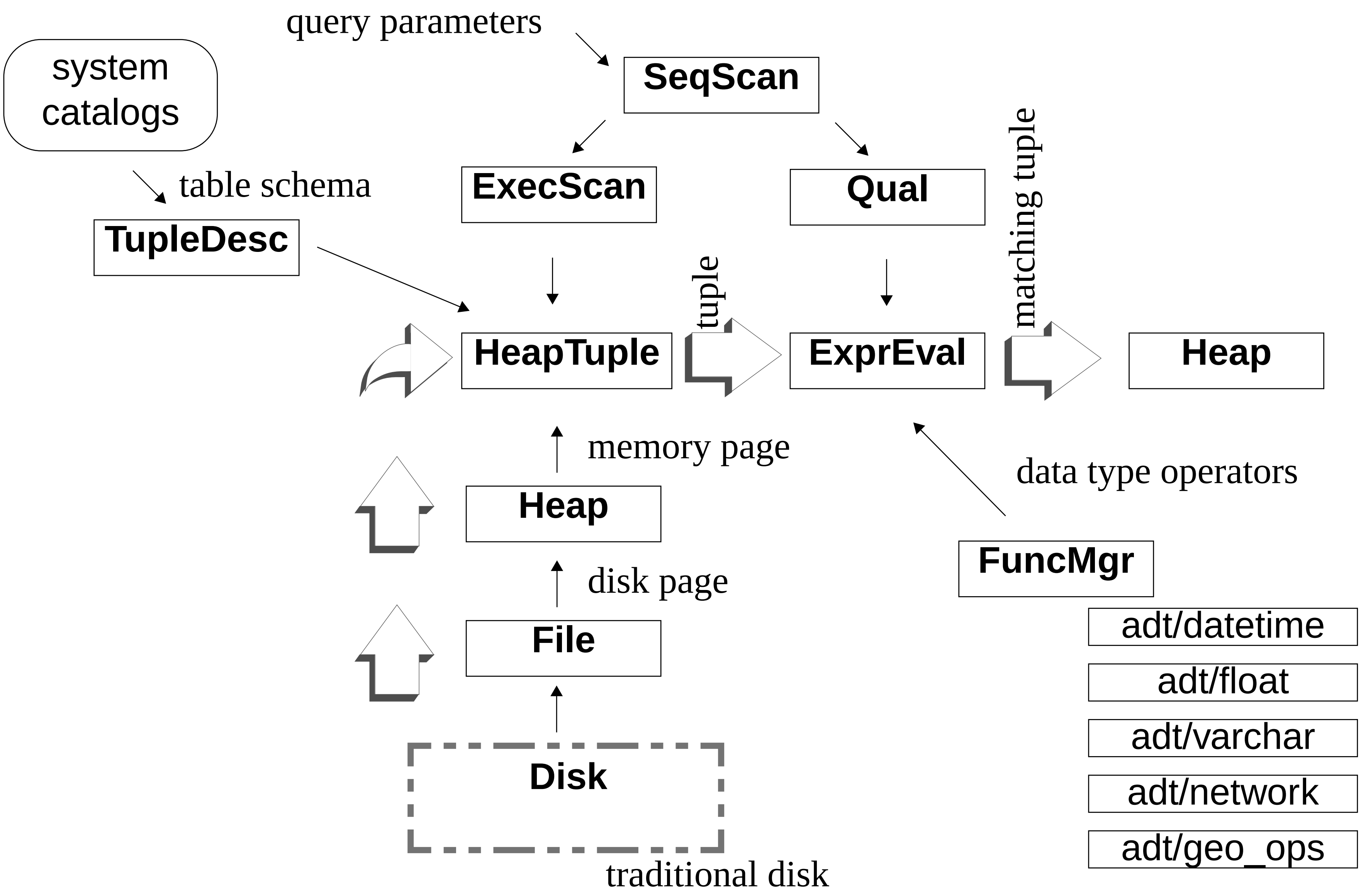
MergeJoin

NestLoop

Agg

Sort

Execute Node



query parameters

system catalogs

SeqScan

table schema

ExecScan

Qual

TupleDesc

matching tuple

tuple

HeapTuple

ExprEval

Heap

memory page

data type operators

Heap

FuncMgr

disk page

File

adt/datetime

adt/float

adt/varchar

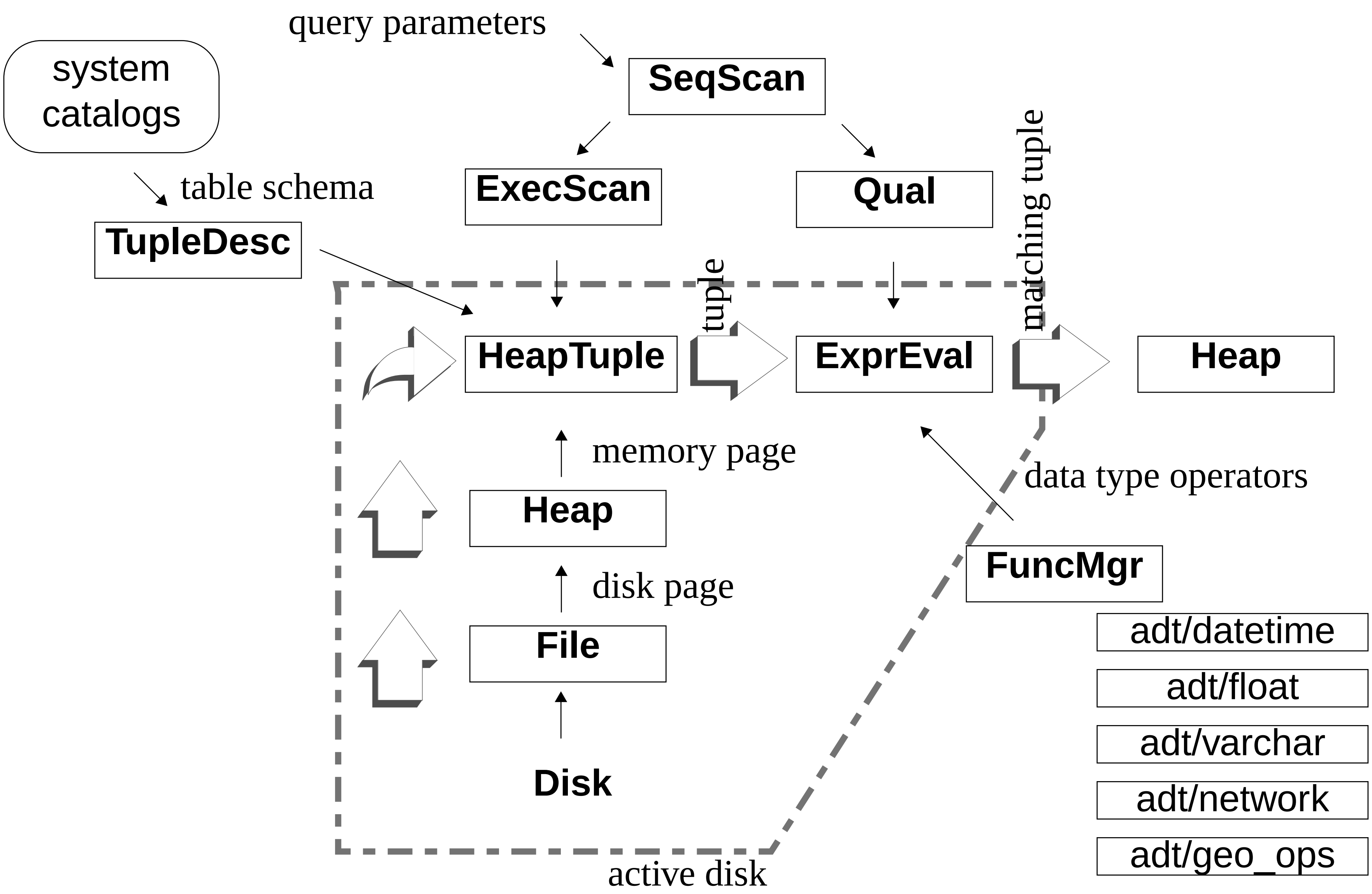
adt/network

adt/geo_ops

Disk

traditional disk

Active Disk Structure



query parameters

system catalogs

SeqScan

table schema

ExecScan

Qual

TupleDesc

tuple

matching tuple

HeapTuple

ExprEval

Heap

memory page

data type operators

Heap

FuncMgr

disk page

File

adt/datetime

adt/float

adt/varchar

adt/network

adt/geo_ops

Disk

active disk

Objections to Active Disks for Database

“Performance benefits are too small”

- claim: parallelism just isn’t there

“Functionality is too complicated for Active Disks”

“Too difficult to change existing code”

“This has been tried before, and didn’t succeed”

- database machines didn’t take over the world

“Can just do it with a bunch of PCs”

- cost argument, not covered here

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

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

```
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)
```

Statistics

```
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)
```

Attributes

Objections to Active Disks for Database

“Performance benefits are too small”

- claim: parallelism just isn’t there

“Functionality is too complicated for Active Disks”

“Too difficult to change existing code”

“This has been tried before, and didn’t succeed”

- database machines didn’t take over the world

“Can just do it with a bunch of PCs”

- cost argument, not covered here

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 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 **- that’s why we’re here**

Objections to Active Disks for Database

- “Performance benefits are too small”
  - claim: parallelism just isn’t there
- “Functionality is too complicated for Active Disks”
- “Too difficult to change existing code”
- “This has been tried before, and didn’t succeed”
  - database machines didn’t take over the world
- “Can just do it with a bunch of PCs”
  - cost argument, not covered here

Conclusions

Significant performance benefits

- for all three basic operations - select, project, join
- 20% to 2.5x in prototype system
- extrapolate 40% to more than 10x in larger systems

Modification of database for Active Disks is feasible

- changed ~2% of the database code
- run ~5% of the total code at the drives
- six person-months effort

Additional benefits possible with on-disk functions

- code specialization
- integrated scheduling