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

<table>
<thead>
<tr>
<th>System</th>
<th>Processing (MHz)</th>
<th>Data Rate (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compaq Proliant TPC-C</td>
<td>4 x 400=1,600</td>
<td>141 x 200=28,200</td>
</tr>
<tr>
<td>Compaq AlphaServer 500 TPC-C</td>
<td>1 x 500=500</td>
<td>61 x 200=12,200</td>
</tr>
<tr>
<td>Compaq AlphaServer 8400 TPC-D</td>
<td>12x612=7,344</td>
<td>521 x 200=104,200</td>
</tr>
<tr>
<td>Microsoft Terraserver</td>
<td>8 x 440=3,520</td>
<td>324 x 200=64,800</td>
</tr>
<tr>
<td>Compaq AlphaServer 8400 TPC-D</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 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
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]

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

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

Scalable throughput

- **speedup** = (#disks)/(host-cpu-speed/disk-cpu-speed)
- (host-cpu/disk-cpu-speed) \(\sim\) 5  (two processor generations)
- **selectivity** = #bytes-input / #bytes-output
### Performance Model

**Application Parameters**
- \( N_{\text{in}} \) = number of bytes processed
- \( N_{\text{out}} \) = number of bytes produced
- \( w \) = cycles per byte
- \( t \) = run time for traditional system
- \( t_{\text{active}} \) = run time for active disk system

**System Parameters**
- \( s_{\text{cpu}} \) = CPU speed of the host
- \( r_d \) = disk raw read rate
- \( r_n \) = disk interconnect rate

**Active Disk Parameters**
- \( s_{\text{cpu}}' \) = CPU speed of the disk
- \( r_d' \) = active disk raw read rate
- \( r_n' \) = active disk interconnect rate

### Traditional vs. Active Disk Ratios

- \( \alpha_N = \frac{N_{\text{in}}}{N_{\text{out}}} \)
- \( \alpha_d = \frac{r_d'}{r_d} \)
- \( \alpha_n = \frac{r_n'}{r_n} \)
- \( \alpha_s = \frac{s_{\text{cpu}}'}{s_{\text{cpu}}} \)

**Traditional server:**

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

Throughput:
\[
\frac{N_{\text{in}}}{t} = \min\left( d \cdot r_d, r_n, \frac{s_{\text{cpu}}}{w} \right)
\]

**Active Disks:**

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

Throughput active:
\[
\frac{N_{\text{in}}}{t_{\text{active}}} = \min\left(d \cdot r_d', r_n', \frac{N_{\text{in}}}{N_{\text{out}}}, d \cdot \frac{s_{\text{cpu}}'}{w} \right)
\]

Rewriting yields:

Throughput active:
\[
\min\left( \alpha_d \cdot (d \cdot r_d), \alpha_N \cdot \alpha_n \cdot (r_n), d \cdot \alpha_s \cdot \left( \frac{s_{\text{cpu}}}{w} \right) \right)
\]

**Speedup:**

\[
S = \frac{(r_n' \cdot \alpha_N')}{\min(r_n', \frac{s_{\text{cpu}}'}{w})}
\]

For \( 1/\alpha_s < d < \alpha_N \), the speedup is:
\[
S = \frac{d \cdot (s_{\text{cpu}}'/w)}{\min(r_n', s_{\text{cpu}}'/w)}
\]

and for \( d > \alpha_N \), is:
\[
S = \max\left( \alpha_N \cdot \alpha_n, \alpha_N \cdot \alpha_s \cdot \left( \frac{w \cdot r_n'}{s_{\text{cpu}}'} \right) \right) > \alpha_N \cdot \max(\alpha_n, \alpha_s)
\]
Amdahl’s Law

\[
\text{serial} = S
\]

\[
\text{parallel} = \frac{(1 - p) \cdot S + \frac{p \cdot S}{n}}{S}
\]

Speedup in a Parallel System

- \( p \) is parallel fraction
- \((1 - p)\) serial fraction is not improved
Modified Performance Model

Traditional server:

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

Active Disks:

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

- 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

<table>
<thead>
<tr>
<th>query</th>
<th>entire query</th>
<th>scan only</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>input (MB)</td>
<td>result (KB)</td>
</tr>
<tr>
<td>Q1</td>
<td>672</td>
<td>0.2</td>
</tr>
<tr>
<td>Q5</td>
<td>857</td>
<td>0.09</td>
</tr>
<tr>
<td>Q7</td>
<td>857</td>
<td>0.02</td>
</tr>
<tr>
<td>Q9</td>
<td>976</td>
<td>6.5</td>
</tr>
<tr>
<td>Q11</td>
<td>117</td>
<td>0.3</td>
</tr>
</tbody>
</table>

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)

Throughput (MB/s) vs. Number of Disks

Aggregation Q1 (Group By)

Throughput (MB/s) vs. Number of Disks

Two-Way Join

Throughput (MB/s) vs. Number of Disks

Join Q9

Throughput (MB/s) vs. Number of 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”
  • 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

1. Query flows from the top to the bottom of the diagram.
2. The query is parsed by the Parser module.
3. The parser then queries the system catalogs for table statistics.
4. The Traffic Cop module handles the traffic between different parts of the software.
5. The Optimizer module estimates the best paths for execution.
6. The Plan module selects the optimal plan based on the best path.
7. The Execute module processes the chosen plan, using nodes like HashJoin, SeqScan, IndexScan, Group, Unique, MergeJoin, NestLoop, Agg, and Sort.
Active Disk Structure

- SeqScan
- ExecScan
- Qual
- TupleDesc
- HeapTuple
- ExprEval
- Heap
- FuncMgr
- File
- Disk
- system catalogs
- query parameters
- table schema
- tuple
- matching tuple
- data type operators
- adt/datetime
- adt/float
- adt/varchar
- adt/network
- adt/geo_ops
- 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

<table>
<thead>
<tr>
<th>Module</th>
<th>Original</th>
<th>Modified Host (New &amp; Changed)</th>
<th>Active Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Files</td>
<td>Code</td>
<td>Files</td>
</tr>
<tr>
<td>access</td>
<td>72</td>
<td>26,385</td>
<td>-</td>
</tr>
<tr>
<td>bootstrap</td>
<td>2</td>
<td>1,259</td>
<td>-</td>
</tr>
<tr>
<td>catalog</td>
<td>43</td>
<td>13,584</td>
<td>-</td>
</tr>
<tr>
<td>commands</td>
<td>34</td>
<td>11,635</td>
<td>-</td>
</tr>
<tr>
<td>executor</td>
<td>49</td>
<td>17,401</td>
<td>9</td>
</tr>
<tr>
<td>parser</td>
<td>31</td>
<td>9,477</td>
<td>-</td>
</tr>
<tr>
<td>lib</td>
<td>35</td>
<td>7,794</td>
<td>-</td>
</tr>
<tr>
<td>nodes</td>
<td>24</td>
<td>13,092</td>
<td>-</td>
</tr>
<tr>
<td>optimizer</td>
<td>72</td>
<td>19,187</td>
<td>-</td>
</tr>
<tr>
<td>port</td>
<td>5</td>
<td>514</td>
<td>-</td>
</tr>
<tr>
<td>regex</td>
<td>12</td>
<td>4,665</td>
<td>-</td>
</tr>
<tr>
<td>rewrite</td>
<td>13</td>
<td>5,462</td>
<td>-</td>
</tr>
<tr>
<td>storage</td>
<td>50</td>
<td>17,088</td>
<td>1</td>
</tr>
<tr>
<td>tcp</td>
<td>11</td>
<td>4,054</td>
<td>-</td>
</tr>
<tr>
<td>utils/adt</td>
<td>40</td>
<td>31,526</td>
<td>-</td>
</tr>
<tr>
<td>utils/fmgr</td>
<td>4</td>
<td>2,417</td>
<td>-</td>
</tr>
<tr>
<td>utils</td>
<td>81</td>
<td>19,908</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Total</td>
</tr>
<tr>
<td></td>
<td>578</td>
<td>205,448</td>
<td>10</td>
</tr>
</tbody>
</table>

|                |          |                               | 15          |       |
|                |          |                               | Total       |       |
|                |          |                               | 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

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

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

#### Attributes

<table>
<thead>
<tr>
<th>attrelid</th>
<th>attname</th>
<th>atttypid</th>
<th>attdisbursion</th>
<th>attlen</th>
<th>attlen</th>
<th>attnum</th>
</tr>
</thead>
<tbody>
<tr>
<td>18663</td>
<td>l_orderkey</td>
<td>23</td>
<td>2.33122e-06</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>18663</td>
<td>l_partkey</td>
<td>23</td>
<td>1.06588e-05</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>18663</td>
<td>l_suppkey</td>
<td>23</td>
<td>0.000213367</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>18663</td>
<td>l_linenumber</td>
<td>23</td>
<td>0.0998572</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>18663</td>
<td>l_quantity</td>
<td>701</td>
<td>0.00434997</td>
<td>8</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>18663</td>
<td>l_extendedprice</td>
<td>701</td>
<td>2.66427e-06</td>
<td>8</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>18663</td>
<td>l_discount</td>
<td>701</td>
<td>0.0247805</td>
<td>8</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>18663</td>
<td>l_tax</td>
<td>701</td>
<td>0.0321099</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>18663</td>
<td>l_returnflag</td>
<td>1042</td>
<td>0.307469</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18663</td>
<td>l_linenumber</td>
<td>1042</td>
<td>0.300911</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18663</td>
<td>l_extendedprice</td>
<td>1082</td>
<td>8.94076e-05</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18663</td>
<td>l_discounrate</td>
<td>1082</td>
<td>8.33926e-05</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18663</td>
<td>l_quantity</td>
<td>1082</td>
<td>8.90733e-05</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18663</td>
<td>l_tax</td>
<td>1042</td>
<td>0.100238</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18663</td>
<td>l_returnflag</td>
<td>1042</td>
<td>0.0451101</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18663</td>
<td>l_linenumber</td>
<td>1042</td>
<td>0</td>
<td>16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[...more...]

(572 rows)
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
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