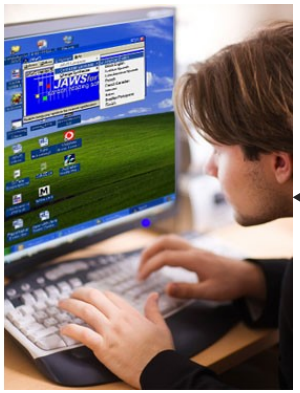
Design Implications for Enterprise Storage Systems via Multi-Dimensional Trace Analysis

Yanpei Chen, Kiran Srinivasan, Garth Goodson, Randy Katz

UC Berkeley AMP Lab, NetApp Inc.

Motivation – Understand data access patterns

Client



Server

How do apps access data?

How do users access data?

How are files accessed?

How are directories accessed?

Better insights → better storage system design

Improvements over prior work

- Minimize expert bias
  - Make fewer assumptions about system behavior
- Multi-dimensional analysis
  - Correlate many dimensions to describe access patterns
- Multi-layered analysis
  - Consider different semantic scoping

Example of multi-dimensional insight

Files with >70% sequential read or sequential write have no repeated reads or overwrites.

- Covers 4 dimensions
  1. Read sequentiality
  2. Write sequentiality
  3. Repeated reads
  4. Overwrites
- Why is this useful?
  - Measuring one dimension easier
  - Captures other dimensions for free

Outline

Observe

1. Traces

- Define semantic access layers
- Extract data points for each layer

Analyze

2. Identify access patterns

- Select dimensions, minimize bias
- Perform statistical analysis (kmeans)

Interpret

3. Draw design implications

- Interpret statistical analysis
- Translate from behavior to design

CIFS traces

- Traced CIFS (Windows FS protocol)
- Collected at NetApp datacenter over three months
- One corporate dataset, one engineering dataset
- Results relevant to other enterprise datacenters

Scale of traces

- Corporate production dataset
  - 2 months, 1000 employees in marketing, finance, etc.
  - 3TB active storage, Windows applications
  - **509,076 user sessions**, 138,723 application instances
  - **1,155,099 files**, 117,640 directories
- Engineering production dataset
  - 3 months, 500 employees in various engineering roles
  - 19TB active storage, Windows and Linux applications
  - **232,033 user sessions**, 741,319 application instances
  - **1,809,571 files**, 161,858 directories

Covers several semantic access layers

- Semantic layer
  - Natural scoping for grouping data accesses
  - E.g. a client's behavior ≠ aggregate impact on server
- Client
  - User sessions, application instances
- Server
  - Files, directories
- CIFS allows us to identify these layers
  - Extract client side info from the traces (users, apps)

Outline

Observe

1. Traces

- Define semantic access layers
- Extract data points for each layer

Analyze

2. Identify access patterns

- Select dimensions, minimize bias
- Perform statistical analysis (kmeans)

Interpret

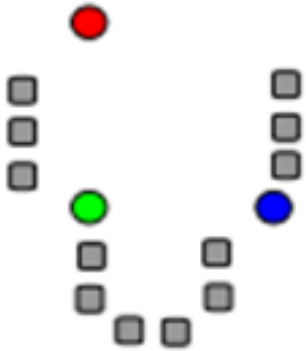
3. Draw design implications

- Interpret statistical analysis
- Translate from behavior to design

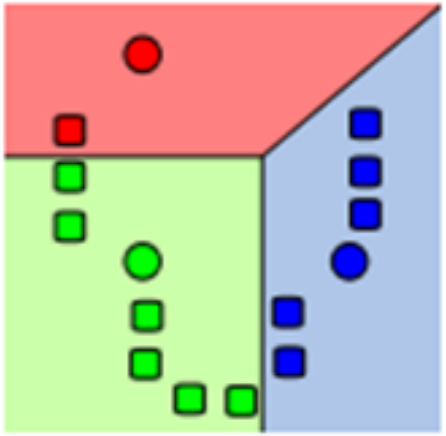
Multi-dimensional analysis

- Many dimensions describe an access pattern
  - E.g. IO size, read/write ratio …
  - Vector across these dimensions is a data point
- Multiple dimensions help minimize bias
  - Bias arises from designer assumptions
  - Assumptions influence choice of dimensions
  - Start with many dimensions, use statistics to reduce
- Discover complex behavior
  - Manual analysis limited to 2 or 3 dimensions
  - Statistical clustering correlates across many dimensions

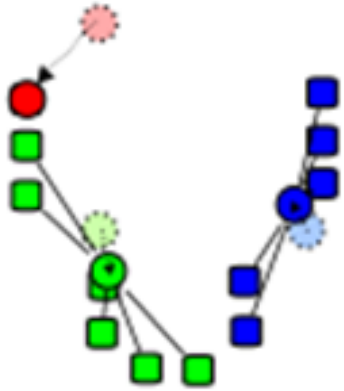
K-means clustering algorithm



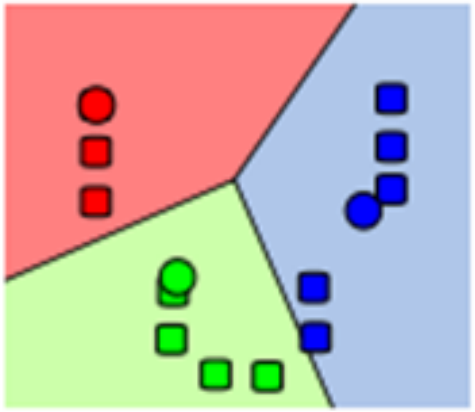
Pick random initial cluster means



Assign multi-D data point to nearest mean



Re-compute means using new clusters



Iterate until the means converge

Cal NetApp

Applying K-means

- For each semantic layer:
  - Pick a large number of relevant dimensions
  - Extract values for each dimension from the trace
  - Run k-means clustering algorithm
  - Interpret resulting clusters
  - Draw design implications

Example – application layer analysis

- Selected 16 dimensions:

1. Total IO size by bytes
2. Read:write ratio by bytes
3. Total IO requests
4. Read:write ratio by requests
5. Total metadata requests
6. Avg. time between IO requests
7. Read sequentiality
8. Write sequentiality
9. Repeated read ratio
10. Overwrite ratio
11. Tree connects
12. Unique trees accessed
13. File opens
14. Unique files opened
15. Directories accessed
16. File extensions accessed

- 16-D data points: 138,723 for corp., 741,319 for eng.
- K-means identified 5 significant clusters for each
- Many dimensions were correlated

Example – application clustering results

Corp. app. instance access patterns	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
% of all app instances	16%	56%	14%	8.8%	5.1%
Total IO	100 KB	0	1 KB	800 KB	3.5 MB
Read:write ratio	1:0	0:0	1:1	1:0	2:3
Metadata requests	130	5	50	130	500
Read sequentiality	5%	-	0%	80%	50%
Write sequentiality	-	-	0%	-	80%
Overwrite ratio	-	-	0%	-	5%
File opens:files	19:4	0:0	10:4	20:4	60:11
Tree connect:Trees	2:2	0:0	2:2	2:2	2:2
Directories accessed	3	0	3	3	4
File extensions accessed	2	0	2	2	3

But what do these clusters mean?

Need additional interpretation …

Outline

Observe

1. Traces

- Define semantic access layers
- Extract data points for each layer

Analyze

2. Identify access patterns

- Select dimensions, minimize bias
- Perform statistical analysis (kmeans)

Interpret

3. Draw design implications

- Interpret statistical analysis
- Translate from behavior to design

Label application types

Corp. app. instance access patterns	Viewing app. gen. content	Supporting metadata	App. gen. file updates	Viewing human gen. content	Content update
% of all app instances	16%	56%	14%	8.8%	5.1%
Total IO	100 KB	0	1 KB	800 KB	3.5 MB
Read:write ratio	1:0	0:0	1:1	1:0	2.3
Metadata requests	130	5	50	130	500
Read sequentiality	5%	-	0%	80%	50%
Write sequentiality	-	-	0%	-	80%
Overwrite ratio	-	-	0%	-	5%
File opens:files	19:4	0:0	10:4	20:4	60:11
Tree connect:Trees	2:2	0:0	2:2	2:2	2:2
Directories accessed	3	0	3	3	4
File extensions accessed	2	0	2	2	3

Design insights based on applications

Corp. app. instance access patterns	Viewing app. gen. content	Supporting metadata	App. gen. file updates	Viewing human gen. content	Content update
% of all app instances	16%	56%	14%	8.8%	5.1%
Total IO	100 KB	0	1 KB	800 KB	3.5 MB
Read:write ratio	1:0	0:0	1:1	1:0	2:3
Metadata requests	130	5	50	130	500
Read sequentiality	5%	-	0%	80%	50%
Write sequentiality	-	-	0%	-	80%
Overwrite ratio	-	-	0%	-	5%
File opens:files	19:4	0:0	10:4	20:4	60:11
Tree connect:Trees	2:2	0:0	2:2	2:2	2:2
Directories accessed	3	0	3	3	4
File extensions accessed	2	0	2	2	3

Observation: Apps with **any** sequential read/write have high sequentiality

Implication: Clients can prefetch based on sequentiality only

Design insights based on applications

Corp. app. instance access patterns	Viewing app. gen. content	Supporting metadata	App. gen. file updates	Viewing human gen. content	Content update
% of all app instances	16%	56%	14%	8.8%	5.1%
Total IO	100 KB	0	1 KB	800 KB	3.5 MB
Read:write ratio	1:0	0:0	1:1	1:0	2:3
Metadata requests	130	5	50	130	500
Read sequentiality	5%	-	0%	80%	50%
Write sequentiality	-	-	0%	-	80%
Overwrite ratio	-	-	0%	-	5%
File opens:files	19:4	0:0	10:4	20:4	60:11
Tree connect:Trees	2:2	0:0	2:2	2:2	2:2
Directories accessed	3	0	3	3	4
File extensions accessed	2	0	2	2	3

Observation: Small IO, open few files multiple times

Implication: Clients should **always** cache the first few KB of every file, in addition to other cache policies

Apply identical method to engineering apps

Eng. app. instance access patterns	Compilation app	Supporting metadata	Content update – small	Viewing human gen. content	Content viewing - small
% of all app instances	1.6%	93%	0.9%	2.0%	2.5%
Total IO	2 MB	0	2 KB	1 MB	3 KB
Read:write ratio	9:1	0:0	0:1	1:0	1:0
Metadata requests	400	1	14	40	15
Read sequentiality	50%	-	-	90%	0%
Write sequentiality	80%	-	0%	-	-
Overwrite ratio	20%	-	0%	-	-
File opens:files	145:75	0:0	3:1	5:4	2:1
Tree connect:Trees	1:1	0:0	1:1	1:1	1:1
Directories accessed	15	0	1	1	1
File extensions accessed	5	0	1	1	1

Identical method can find apps types for other CIFS workloads

Other design insights

Consolidation: Clients can consolidate sessions based on ***only*** the read write ratio.

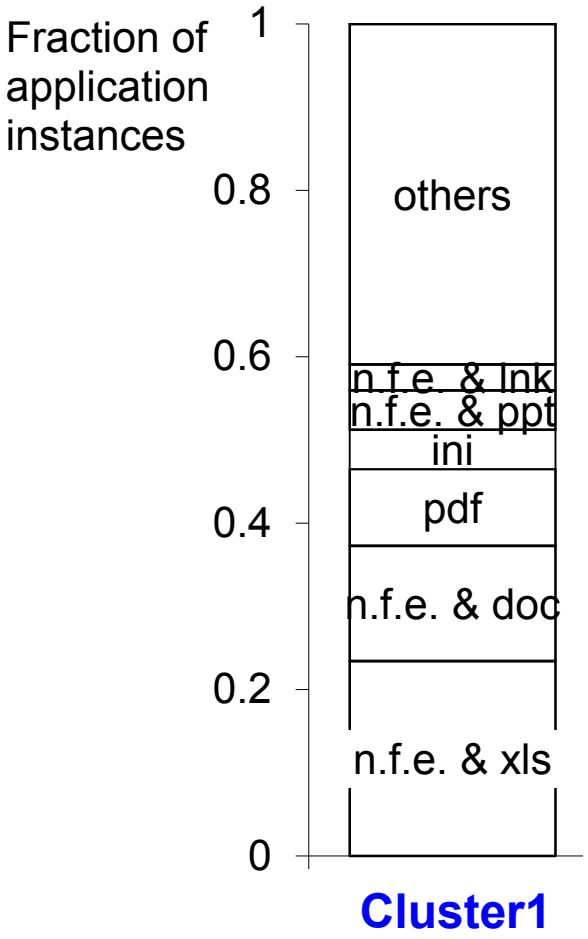
File delegation: Servers should delegate files to clients based on ***only*** access sequentiality.

Placement: Servers can select the best storage medium for each file based on ***only*** access sequentiality.

Simple, threshold-based decisions on one dimension

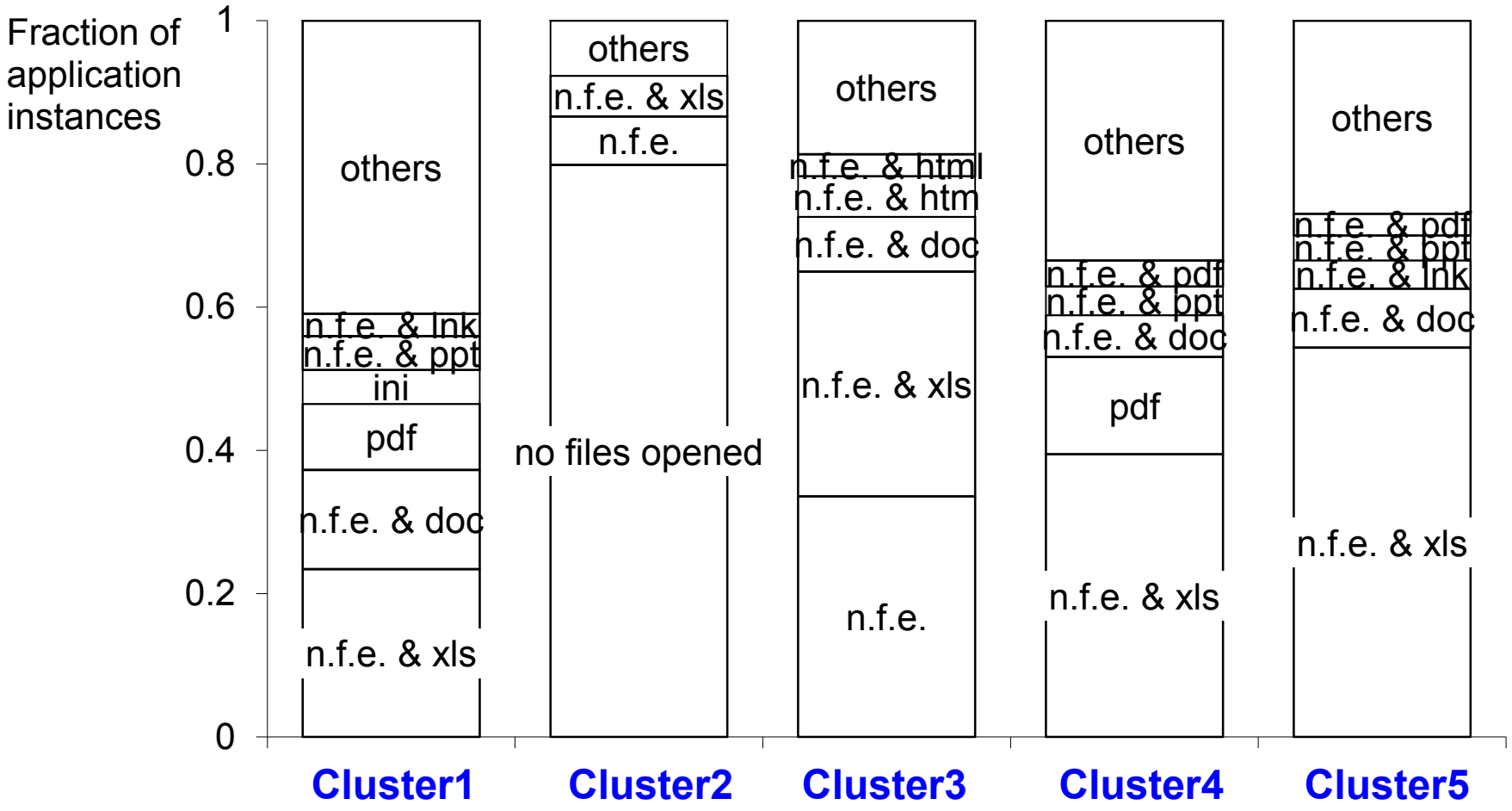
High confidence that it's the correct dimension

New knowledge – app. types depend on IO, not software!



n.f.e. = No file extension

New knowledge – app. types depend on IO, not software!



n.f.e. = No file extension

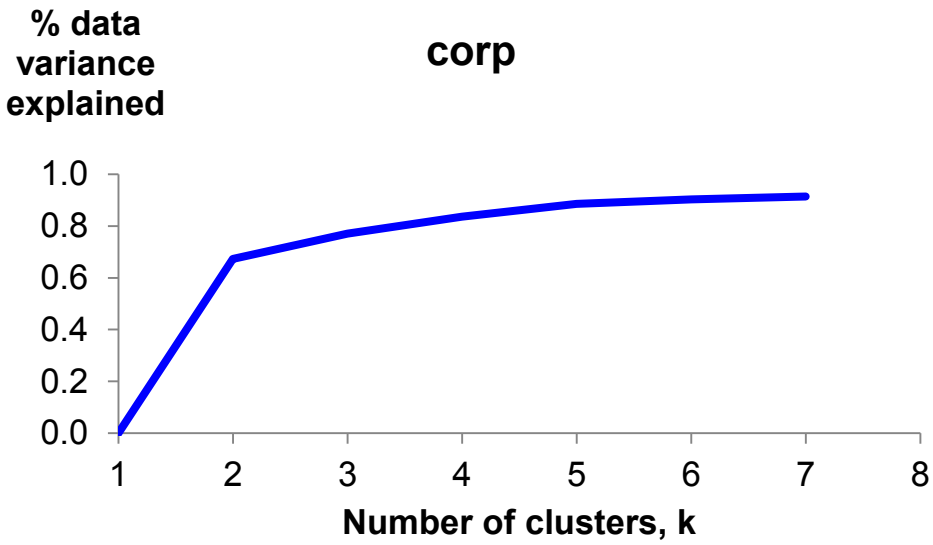
Summary

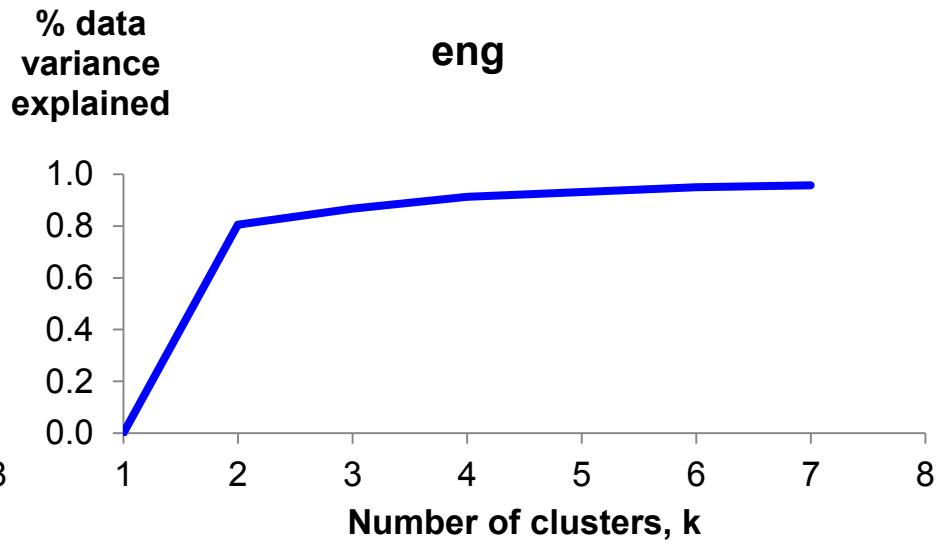
- Contribution:
  - Multi-dimensional trace analysis methodology
  - Statistical methods minimize designer bias
  - Performed analysis at 4 layers – results in paper
  - Derived 6 client and 6 server design implications
- Future work:
  - Optimizations using data content and working set analysis
  - Implement optimizations
  - Evaluate using workload replay tools
- Traces available from NetApp under license

Thanks!!!

Backup slides

How many clusters? – Enough to explain variance





Behavior variation over time