Group-Based Management of Distributed File Caches

Darrell D. E. Long
Ahmed Amer

Storage Systems Research Center
Jack Baskin School of Engineering
University of California, Santa Cruz

Randal Burns

Hopkins Storage Systems Laboratory
Department of Computer Science
Johns Hopkins University

Outline

- Motivation
- The Aggregating Cache
  - Successor prediction and tracking
  - Client Cache performance
- Filtering Effects
  - Server-Side Caching
  - Successor Entropy
  - Visualizing Filtering Effect on Predictability
- Related Work
- Conclusions & Future Work

Motivation

- Improved client & server caching by grouping
  - Reduced miss rates means fewer demand fetches
  - Resilience to client-cache filtering effects
- Avoids pre-fetching drawbacks
  - Incorrect prediction penalties can be limited based on storage system specifications
  - All relationship and prediction maintenance is not time critical

The Aggregating Cache

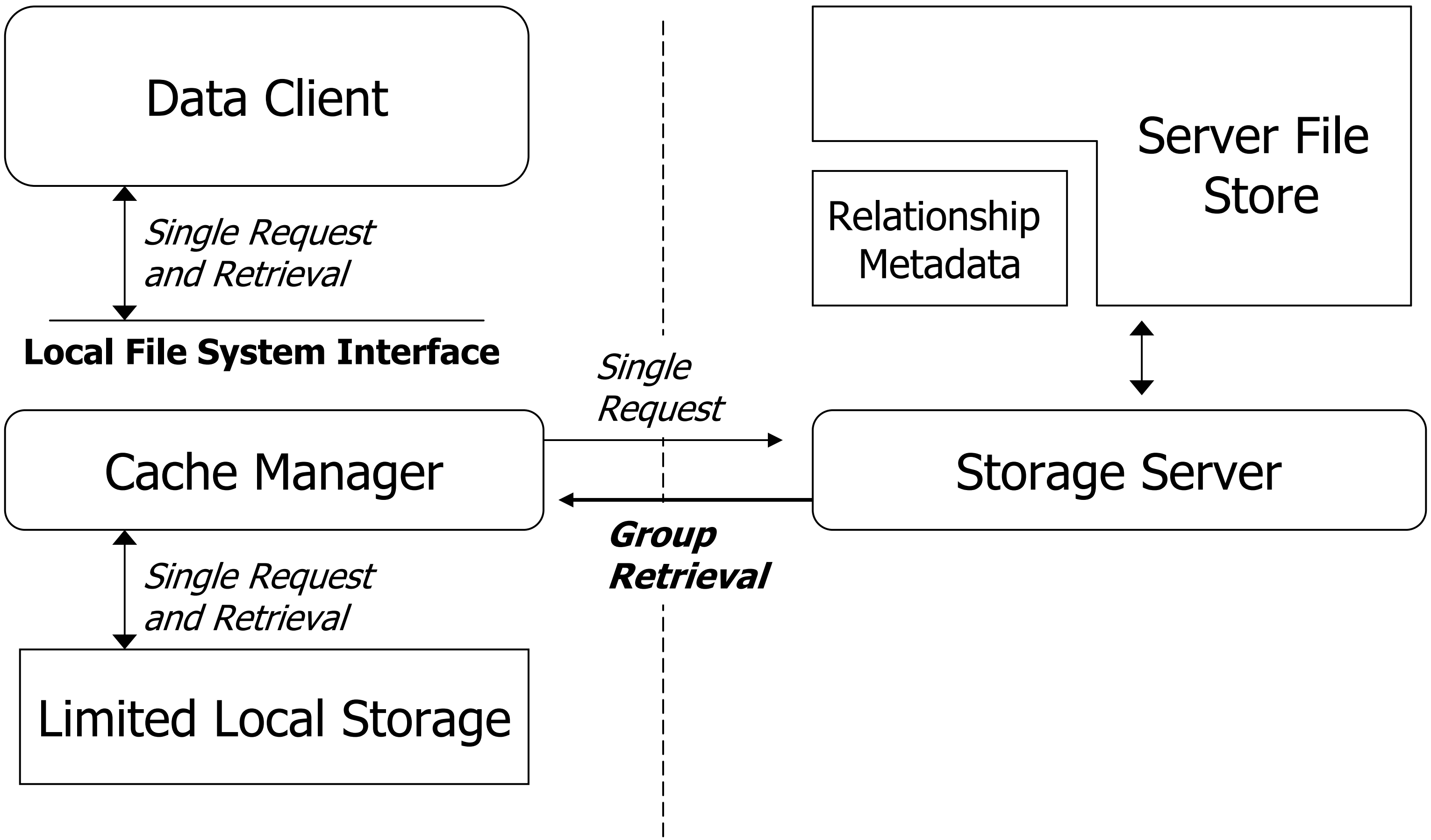
- The aggregating cache is based on the retrieval of pre-built file groups
- Server-maintained groups are ...
  - ... based on file relationship modeling
  - ... pre-constructed at the server
    - This avoids timeliness issues of pre-fetching
  - ... based on an associated set of likely successors for each file

SSRC

The Aggregating Cache (cont'd)

- Groups affect in-cache priority
  - Upon receipt of a request for a file, associated group members are retrieved
    - Files already in the cache need not be retrieved again
- Fewer fetches from the server occur
  - Results in decreased latency
- Group sizes evaluated
  - From 2 to 10 related files (report on groups of 5)

Aggregating Cache



Aggregating Cache (con'td)

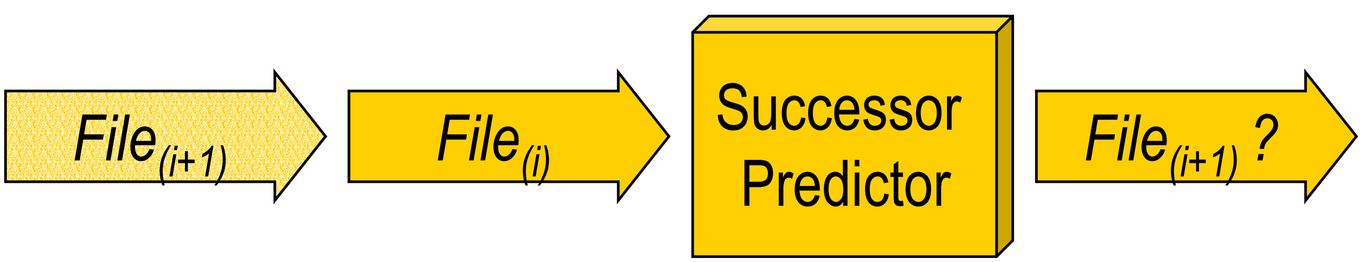
- Do the clients cooperate?
  - Clients gather statistics and forward to the server, or
  - Allow the server to simply observe
- More on this later…

Successor Prediction

- File grouping relies on predictive per-file metadata
- Per-file metadata consists of successor predictions
- Successor predictors are simple, accurate and adjustable
  - Noah
  - Recent popularity

File Successor Prediction

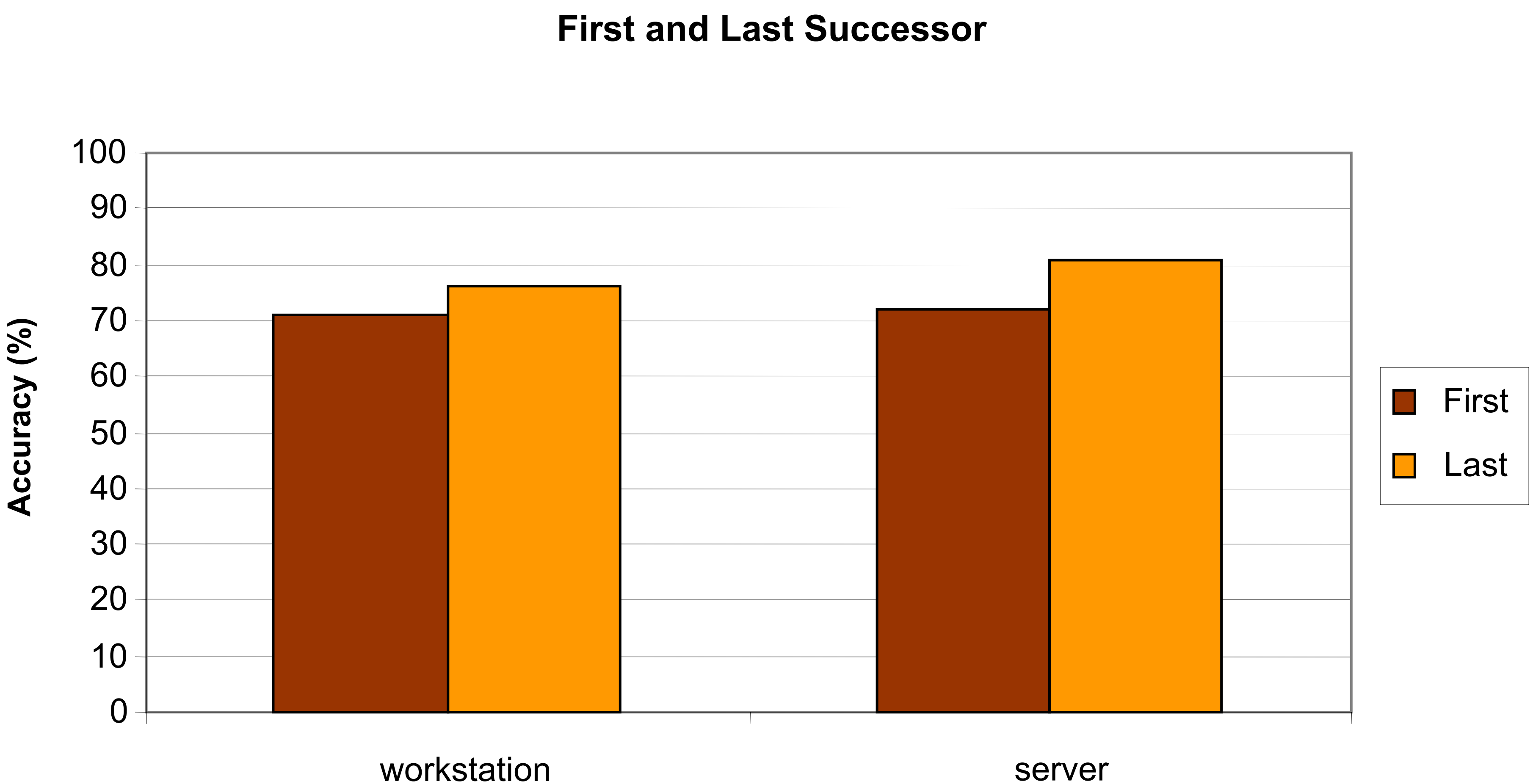
- Given:
  - Observations of the file access stream
  - Knowledge of the current file access
  - Limited per-object state
    - maintainable as file metadata
- Successive file access events are predictable using very simple schemes



Static *vs.* Dynamic Prediction

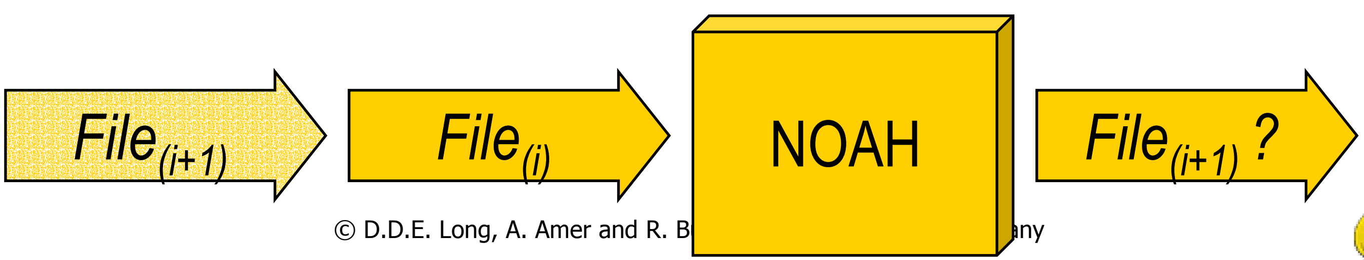
- Static - *First Successor*
  - The file that followed **A** the first time **A** was accessed is always predicted to follow **A**
- Dynamic - *Last Successor*
  - The file that followed **A** the last time **A** was accessed is predicted to follow **A**

Static vs. Dynamic

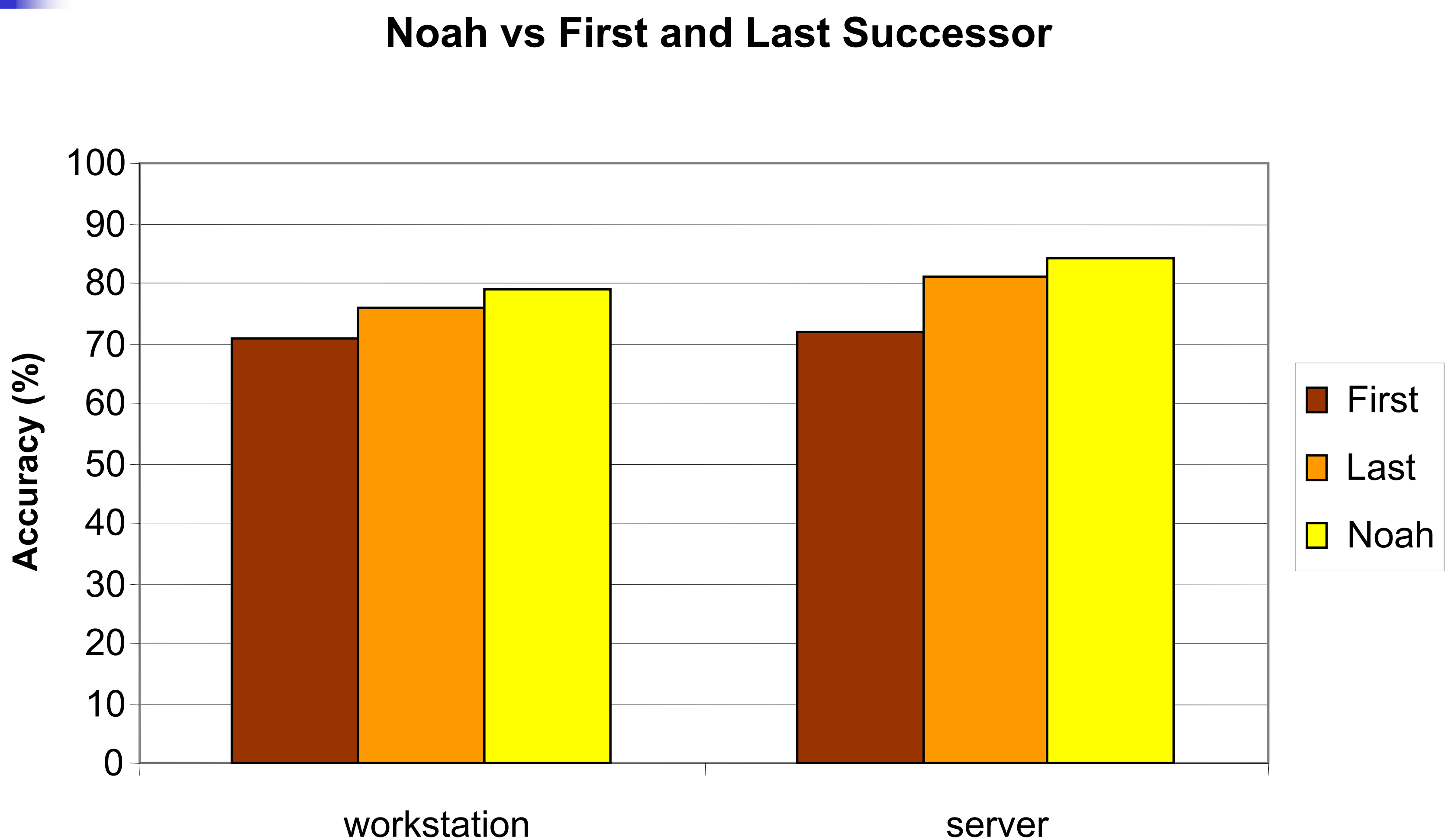


Prediction with Noah

- Last-successor predicts better than first-successor
  - But transient successors cause double-faults for last successor!
- Noah
  - Maintains a current prediction
  - Changes current prediction to last successor if last successor was the same for *S* consecutive accesses
    - *S* (stability) is a parameter, default = 2



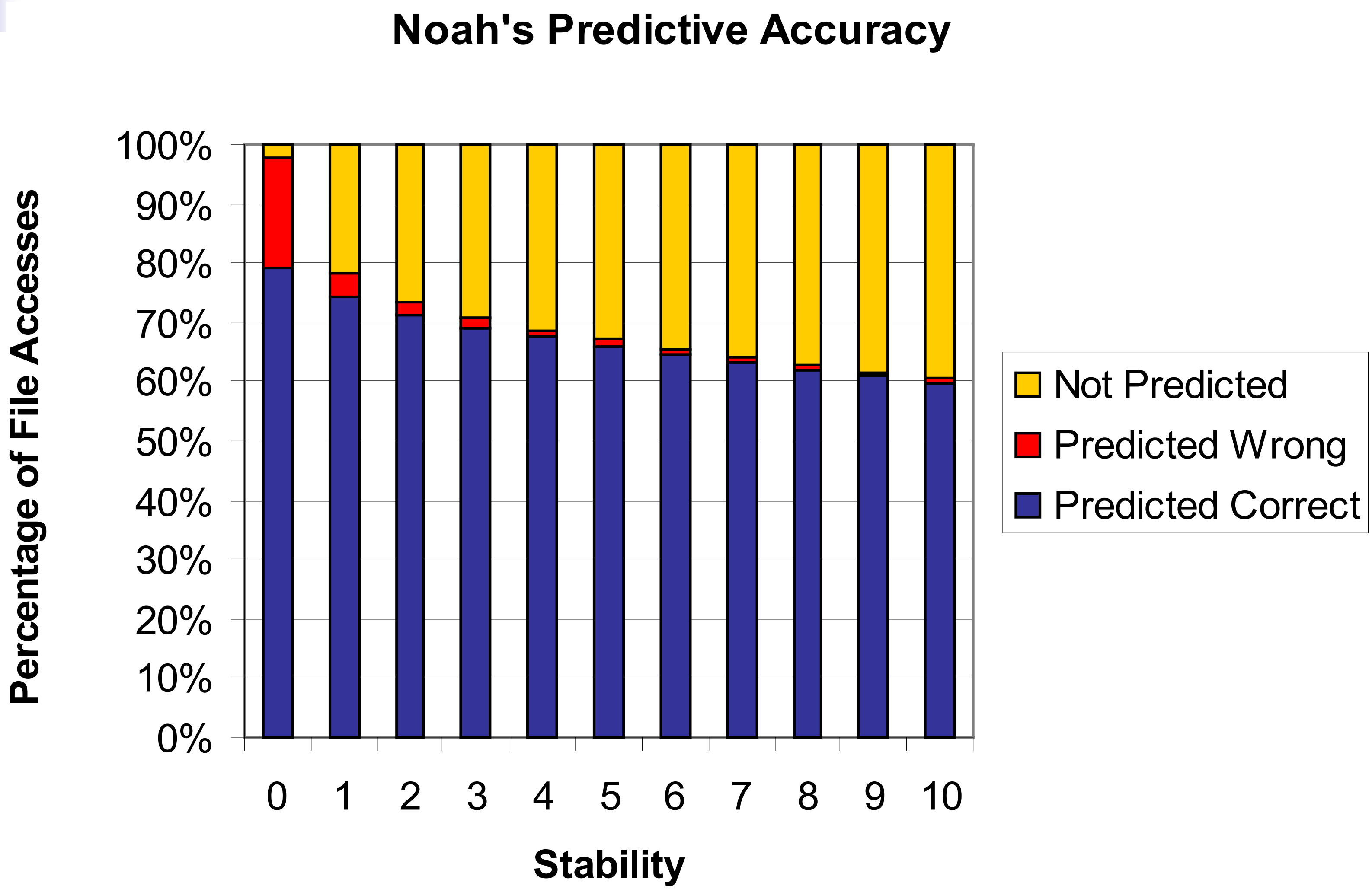
Noah, Static and Dynamic



General and Specific Accuracy

- There is a difference between …
  - … predictor accuracy over a workload
  - … accuracy per prediction
- General Accuracy
  - Percentage of **all events** that are **not predicted or not predicted correctly**
- Specific Accuracy
  - Percentage of all **predictions offered** that are **not correct**

Noah: Varying Stability Parameter



Recent Popularity (Best $_{j \text{ of } k}$)

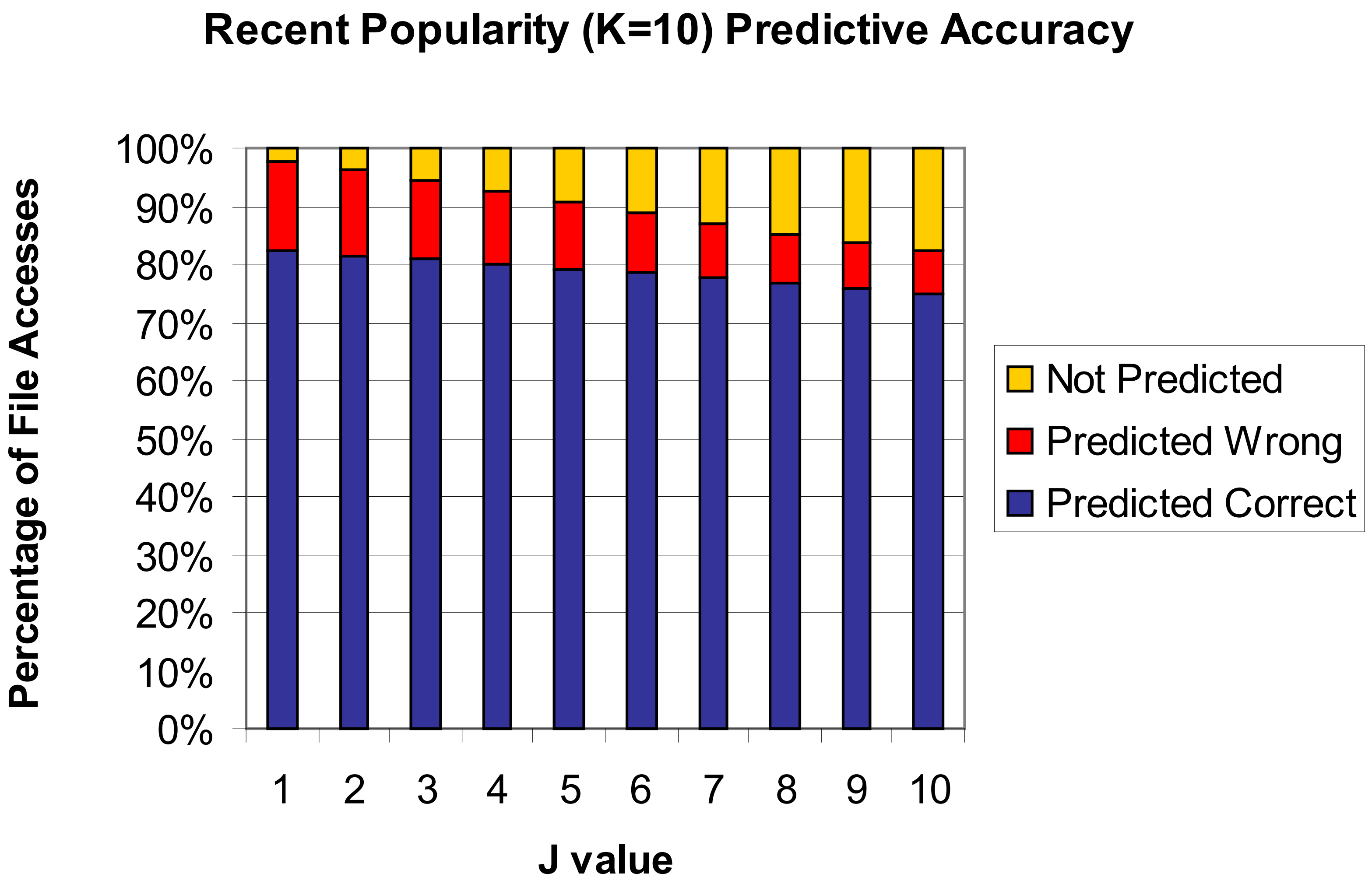
File Access Sequence:

S: A B C D B C D B D B D B C B C B C A B A B A B A B

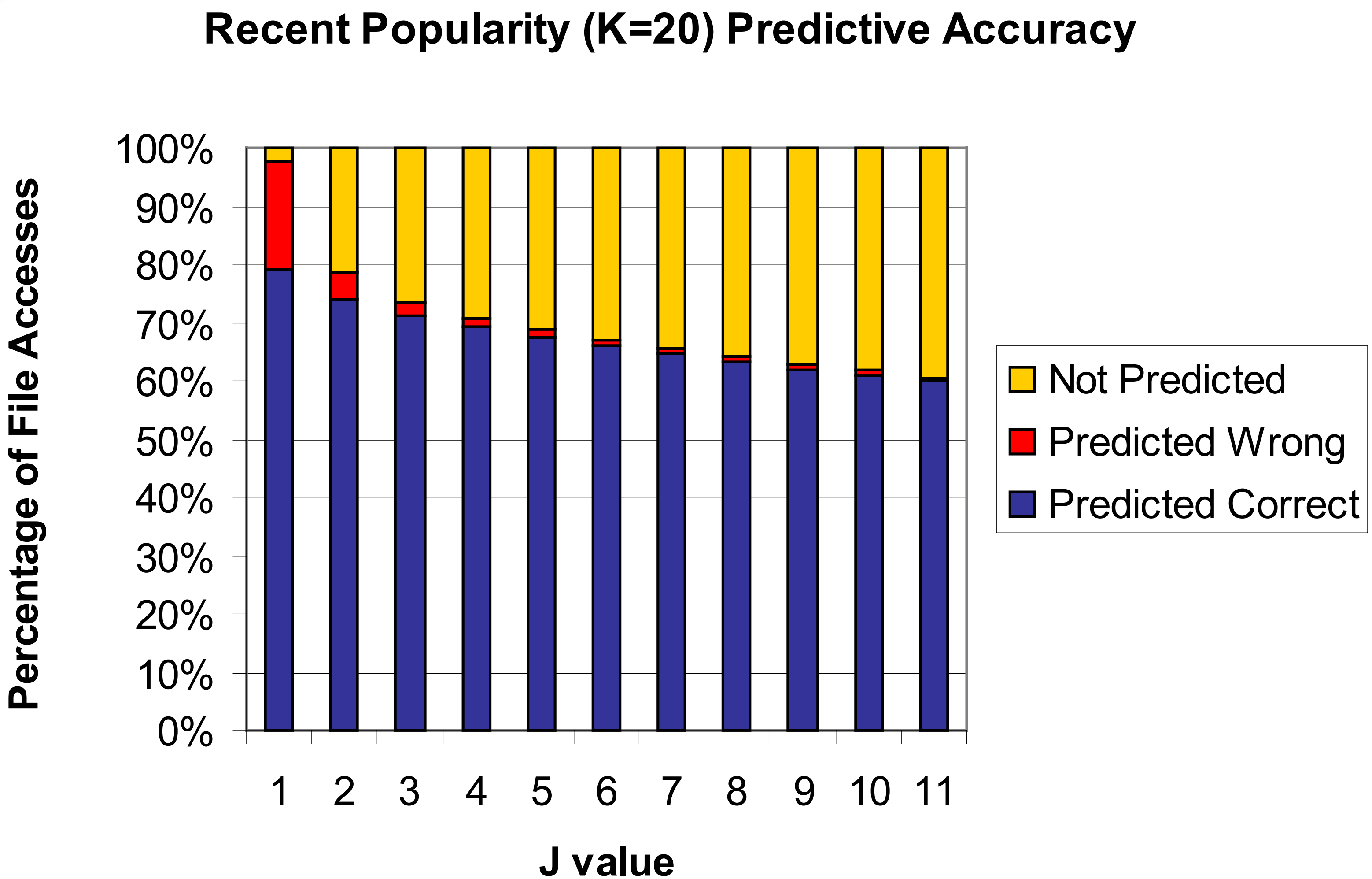
	Per-File Successors	*Successor Counts*
A:	B,B,B,B,B	B:5
B:	C,C,D,D,C,C,C,A,A,A	A:3, C:5, D:2
C:	D,D,B,B,A	A:1, B:2, D:2
D:	B,B,B,B	B:4

$\text{Best}_{(3 \text{ of } 6)}(B|S) = A$
$\text{Best}_{(4 \text{ of } 6)}(B|S) = N/A$
$\text{Best}_{(4 \text{ of } 9)}(B|S) = C$

Recent Popularity (Best $_{j \text{ of } k}$)
Varying J Parameter (K=10)



Recent Popularity (Best $_{j \text{ of } k}$)
Varying J Parameter (K=20)



Successor Prediction

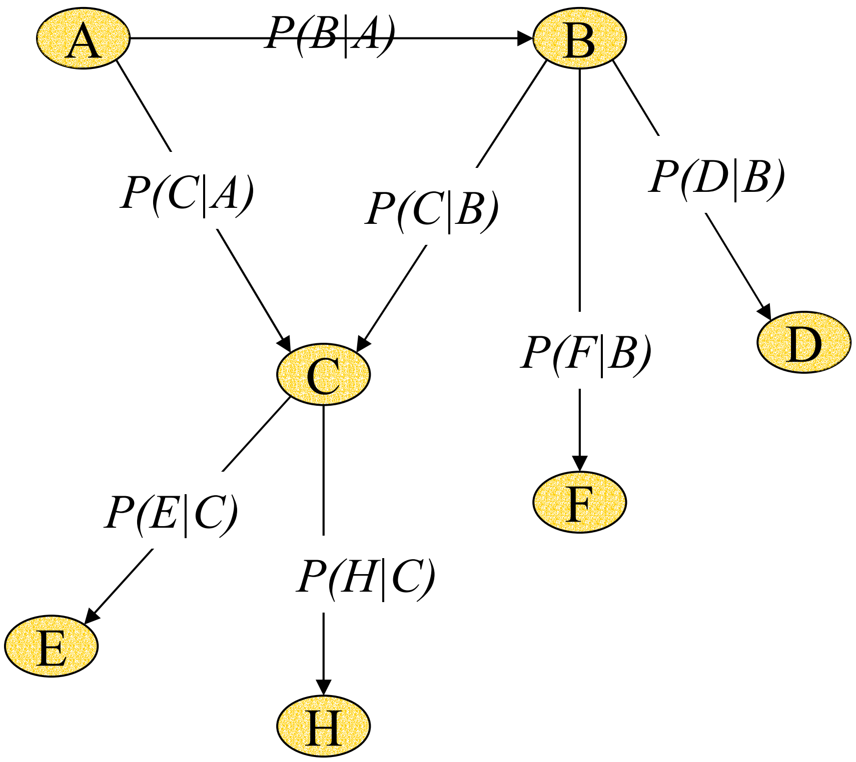
- Static prediction schemes remain valid for extended periods – and for very popular files
- Variation amongst file successors is very limited
- Noah and Recent Popularity are effective and adjustable successor predictors

File Grouping

- Given:
  - Accurate file successor predictions
  - Per-file successor metadata
  - Knowledge of the current file access
- A group of n files can be constructed of those most likely to be accessed in the near future

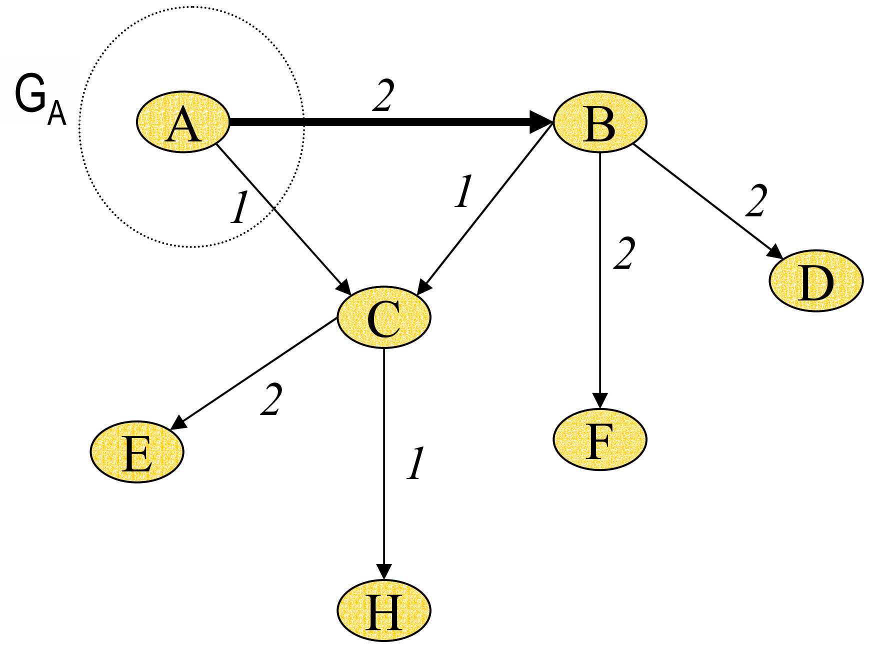
File Relationship Graph

- File successor observations give us probability of a given file following another
  - Fixed set of successors, P(Y|X) $\in$ [0,1,…,S]
- Can construct a *file relationship graph*
  - Nodes: Files
  - Edges: succession probability



Constructing File Groups

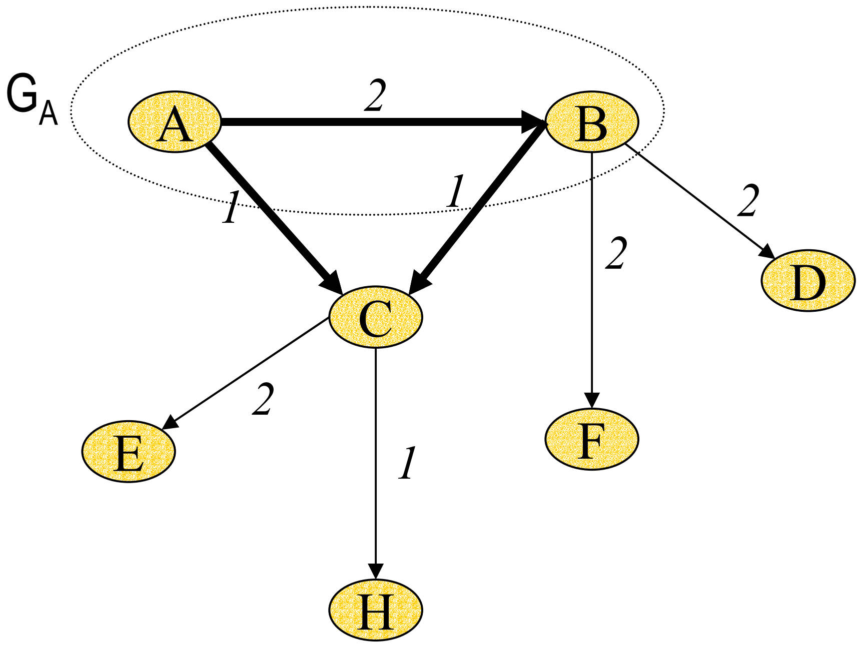
- Given an access to file A, what n files constitute A's group G_A
- *n Best Successor* algorithm
  - $G_A \leftarrow \{A\}$
  - $G_A \leftarrow G_A \cup \{X\}$, for X with maximal $P(X|A)$
  - Repeat until $|G_A| = n$



Example of $\boldsymbol{n} = 3$

Constructing File Groups

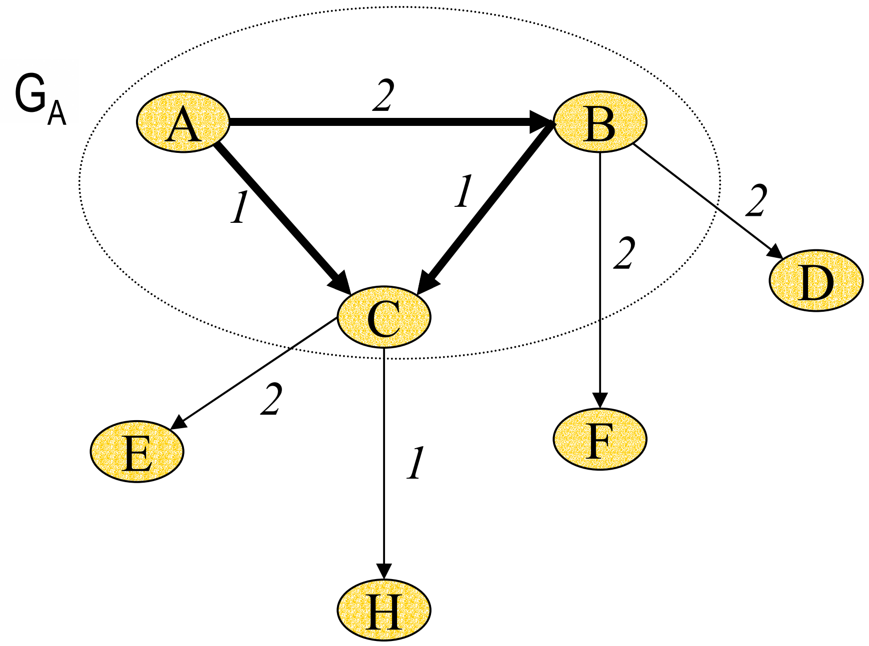
- Given an access to file A, what n files constitute A's group G_A
- *n Best Successor* algorithm
  - $G_A \leftarrow \{A\}$
  - $G_A \leftarrow G_A \cup \{X\}$, for X with maximal P(X|A)
  - Repeat until $|G_A|=n$



Example of $\boldsymbol{n}$ = 3

Constructing File Groups

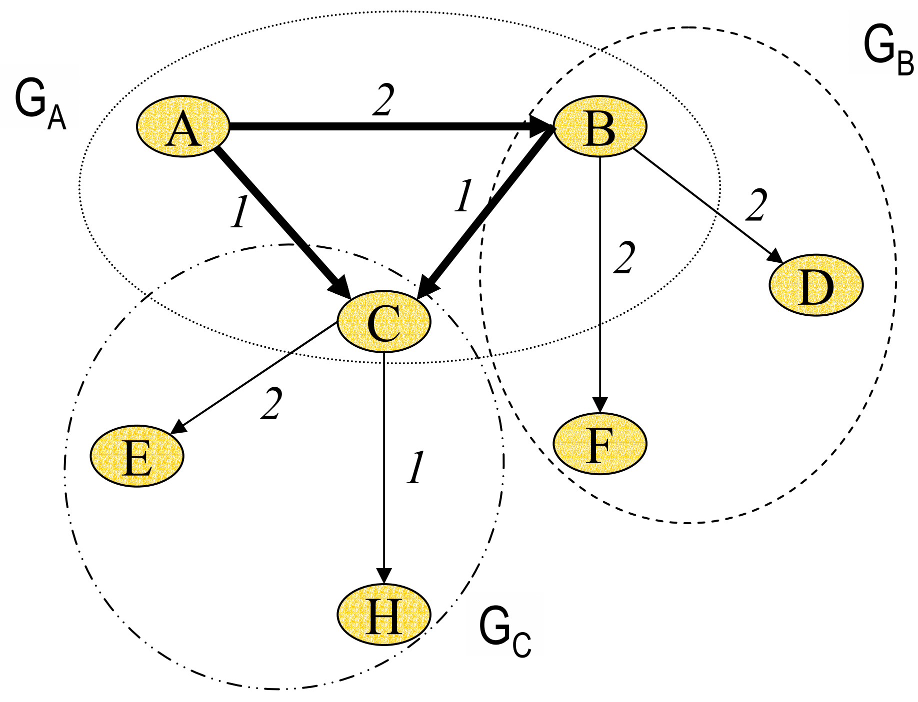
- Given an access to file A, what n files constitute A's group G_A
- n *Best Successor* algorithm
  - $G_A \leftarrow \{A\}$
  - $G_A \leftarrow G_A \cup \{X\}$, for X with maximal $P(X|A)$
  - Repeat until $|G_A| = n$



Example of $\boldsymbol{n} = 3$

Constructing File Groups

- Given an access to file A, what n files constitute A's group G_A
- *n Best Successor* algorithm
  - $G_A \leftarrow \{A\}$
  - $G_A \leftarrow G_A \cup \{X\}$, for X with maximal P(X|A)
  - Repeat until $|G_A|=n$

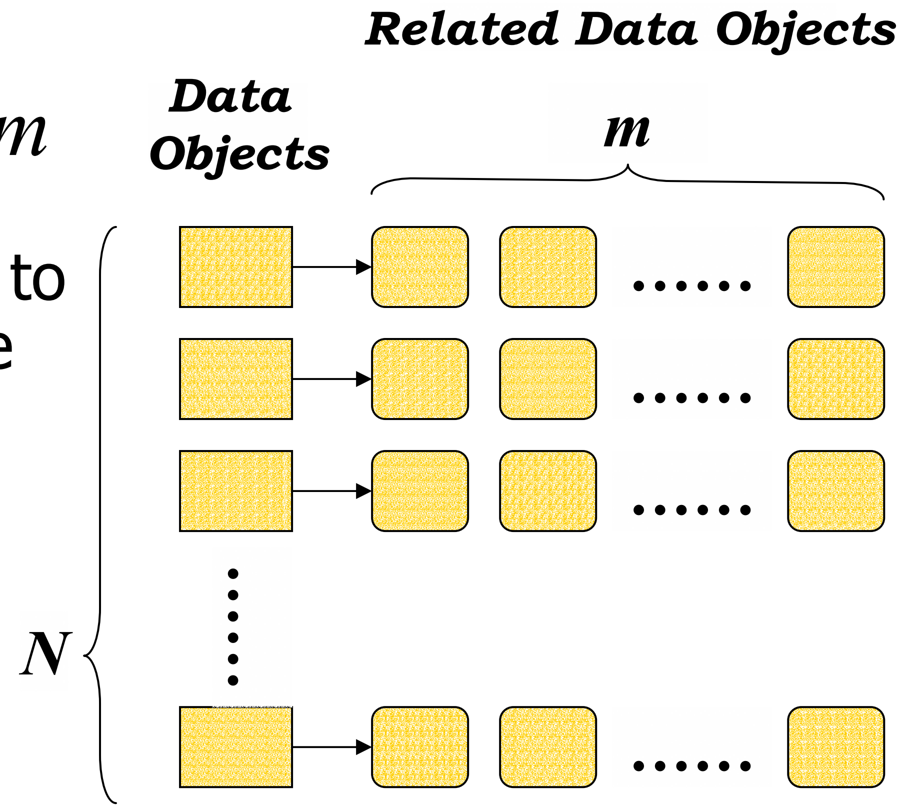


Example of $\boldsymbol{n} = 3$

Server-Maintained Metadata:
A Restricted Relationship Graph

- A simple graph of restricted degree, $\leq m$
- Maximum number of vertices is equivalent to the number of unique files observed in the access stream, N
- Group size $\boldsymbol{n}$

$$n \neq m + 1$$

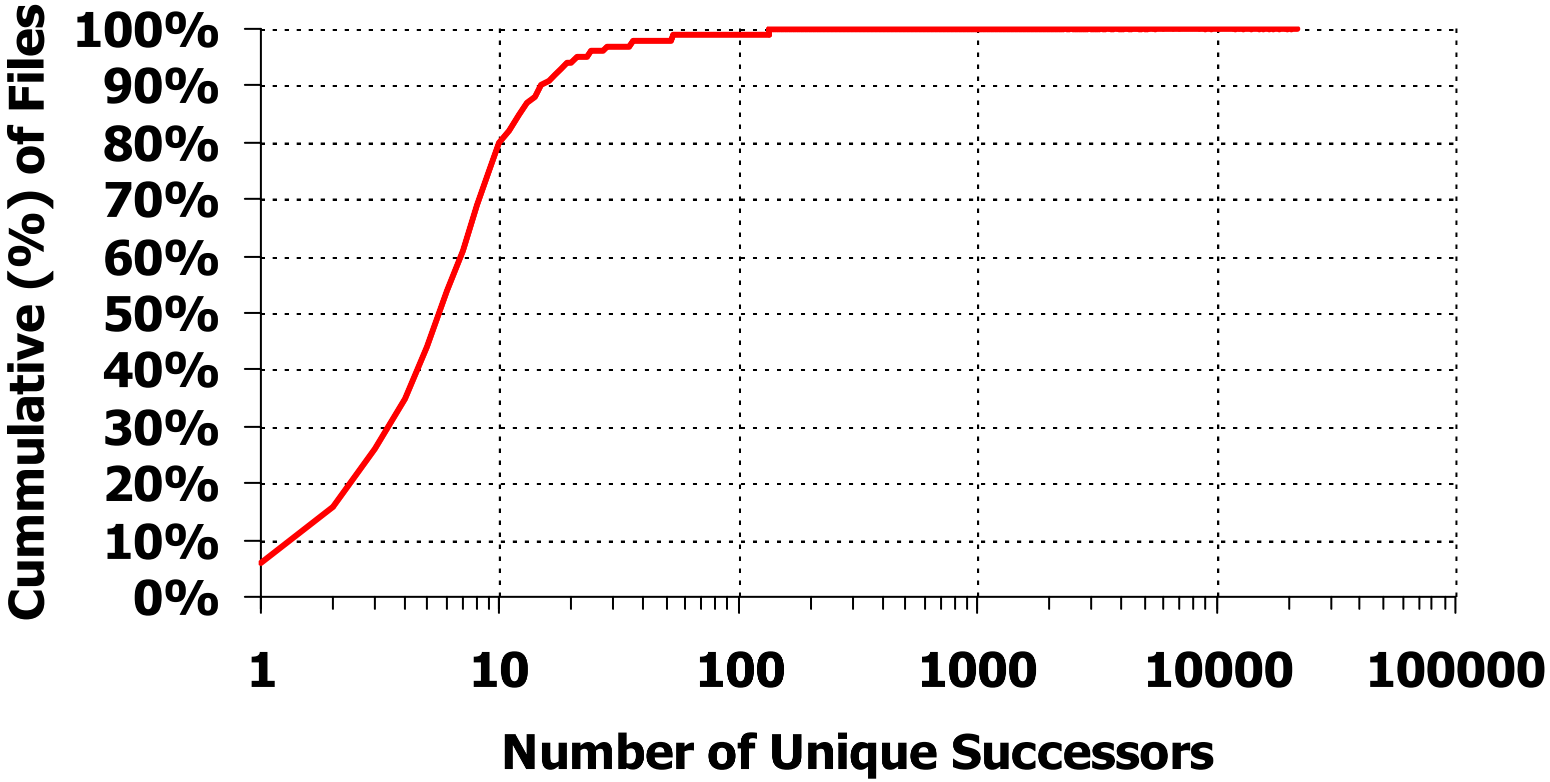


Server-Maintained Metadata

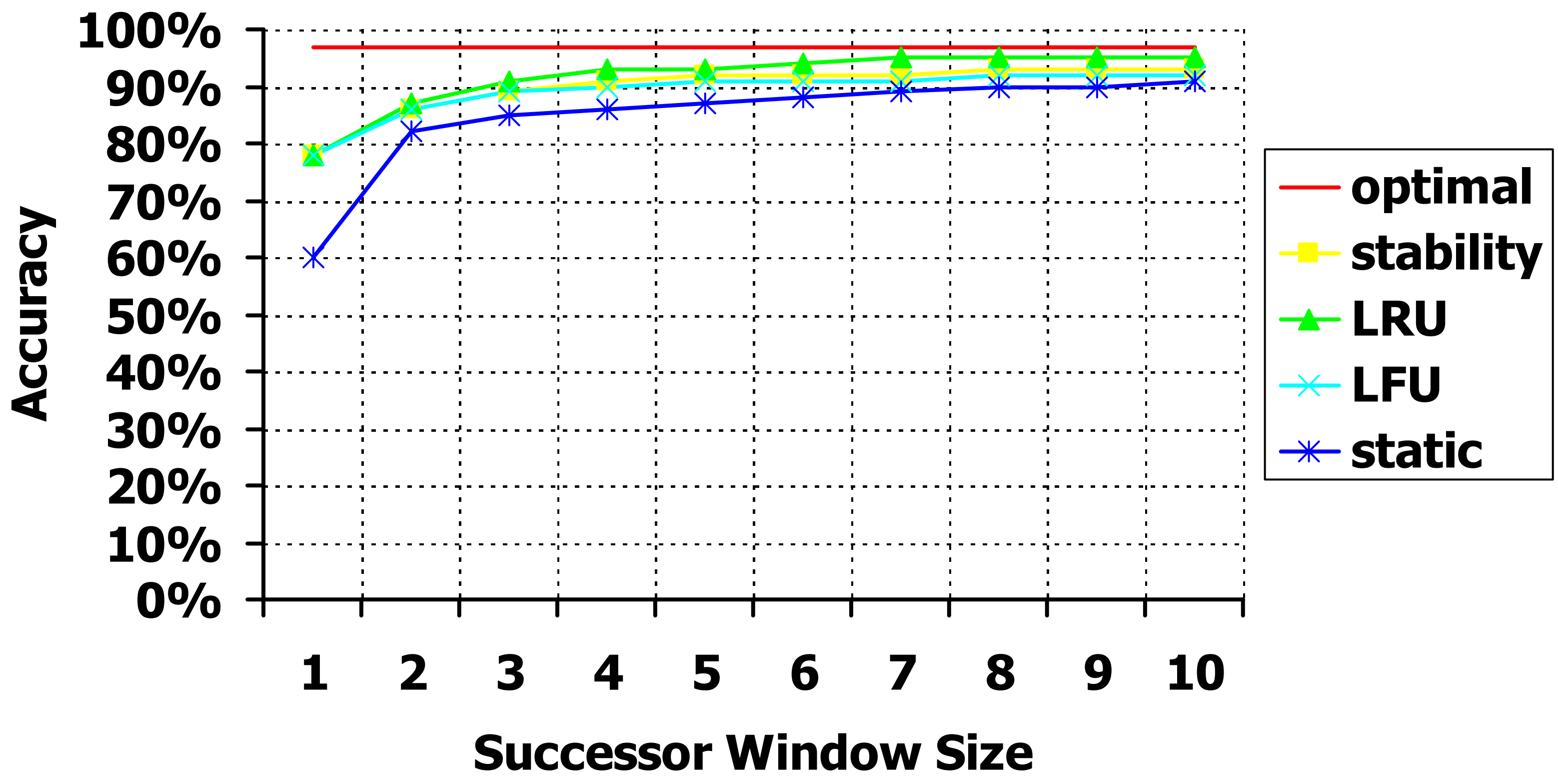
- For each file *A*, we maintain a list of *m* successors S_i and $P(S_i|A)$
- The feasibility of this strategy is dependent on limited variation in file successors
- For our workloads:
  - Over periods of ~1 month, files average less than 10 unique successors
  - Over periods of ~1 year, files average less than 20 unique successors

SSRC

Successor variability

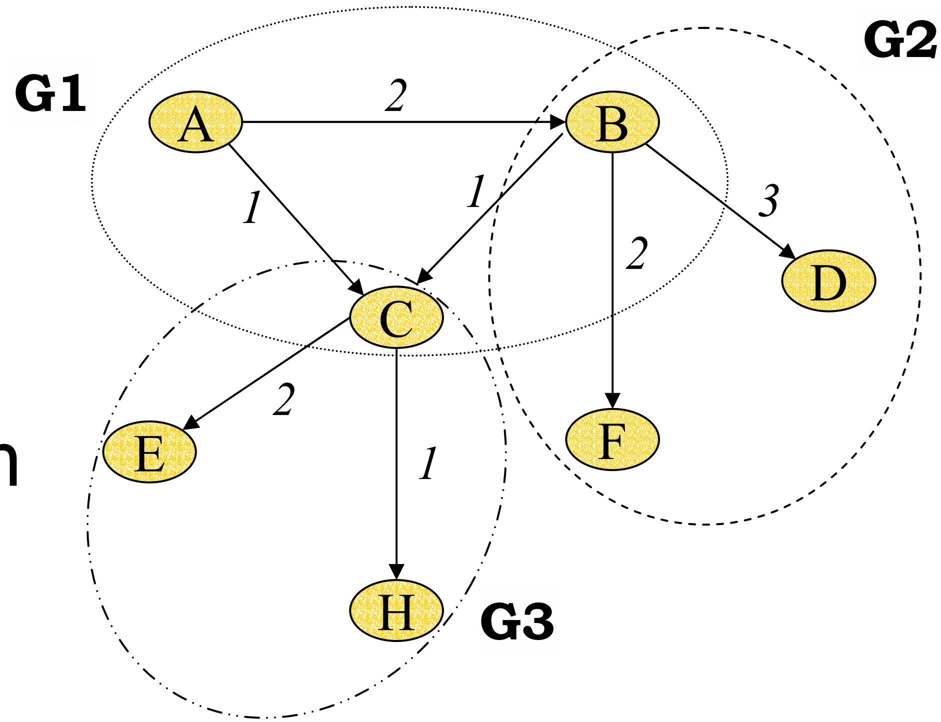


Successor Window Hit Rates



Relationship Graph:
Example Simple Groupings

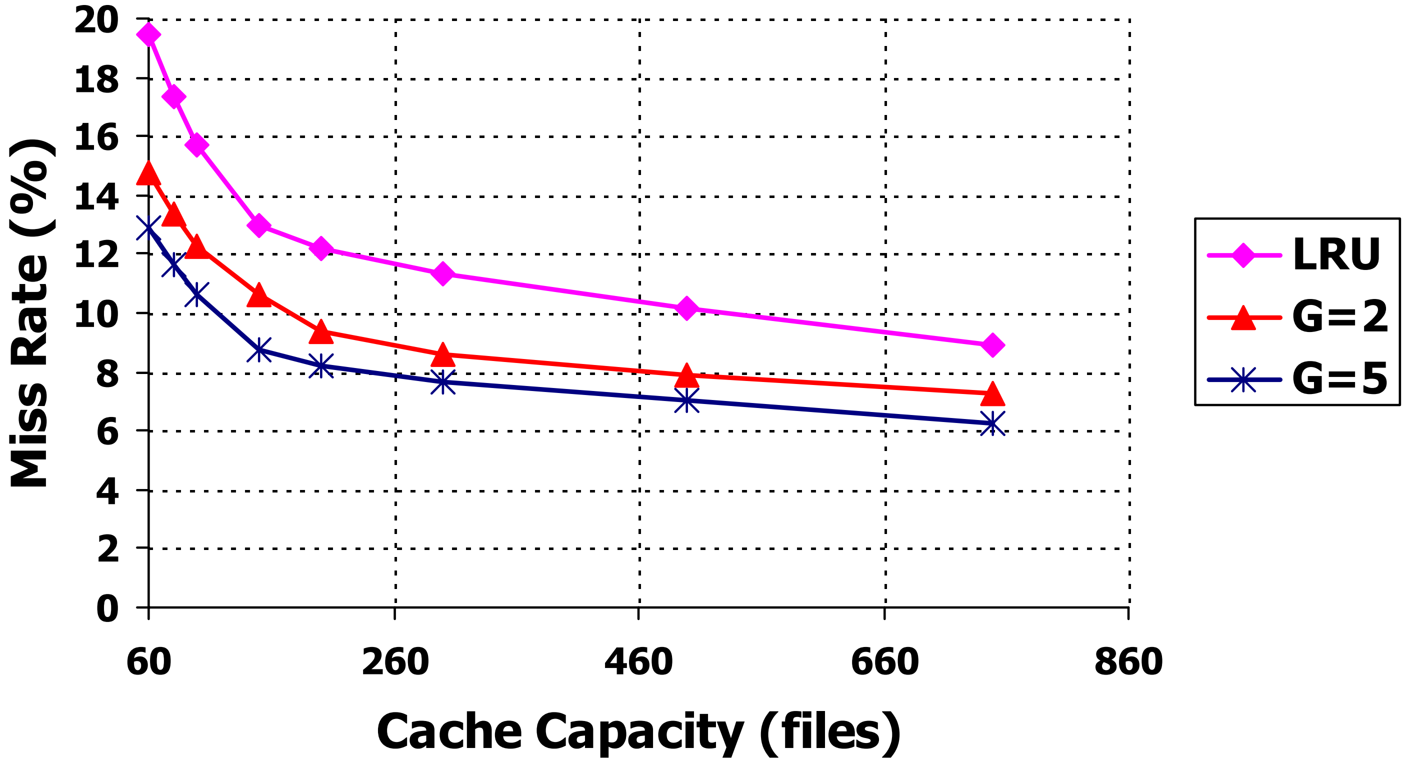
- Groups of size n
- n-1 most likely successors are grouped with each file



Example of $n = 3$

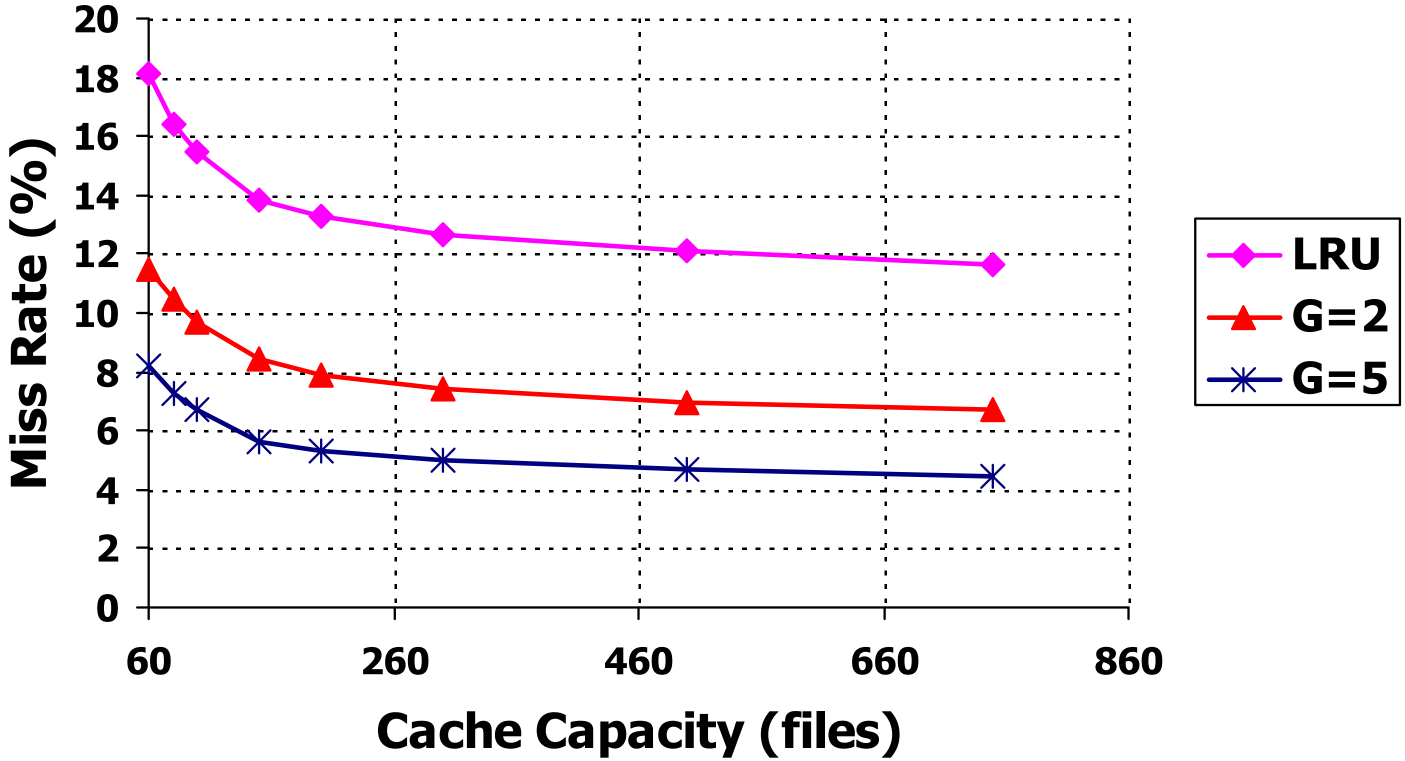
Aggregating Cache
Miss Rates

users **workload**



Aggregating Cache
Miss Rates

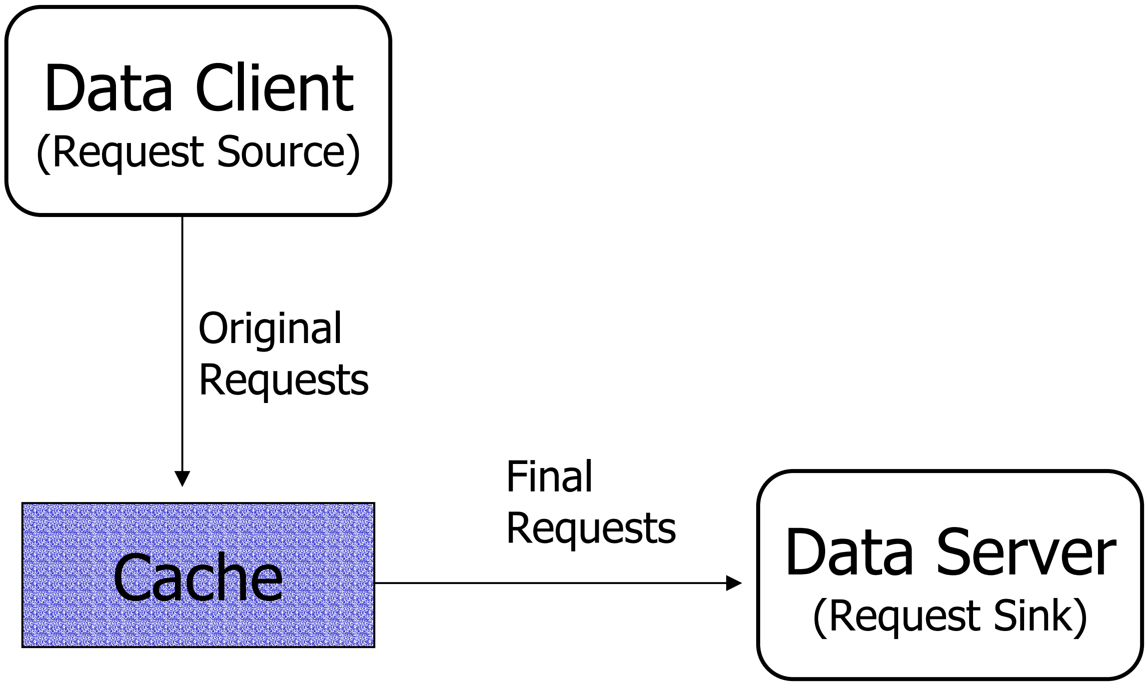
server **workload**



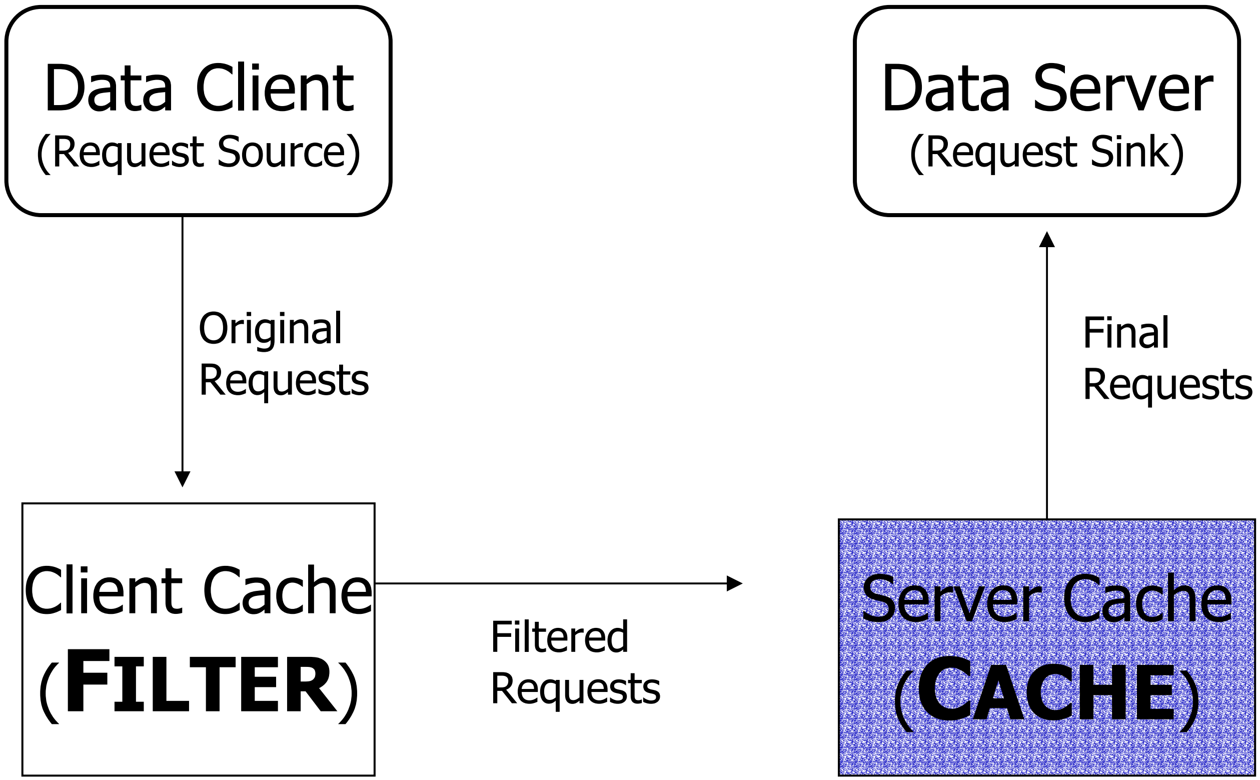
Client Cache Filtering Effects

- Filtered workload
  - Result due to misses from an intervening (client) cache
  - When client and server caches comparable sizes caching can be rendered ineffective for server-side caches
    - Adding a cache is not necessarily a good thing!
- Server-side caching
  - Filtered workloads observed when clients provide no access information beyond cache misses
    - But filtered workloads turn out to be highly predictable!

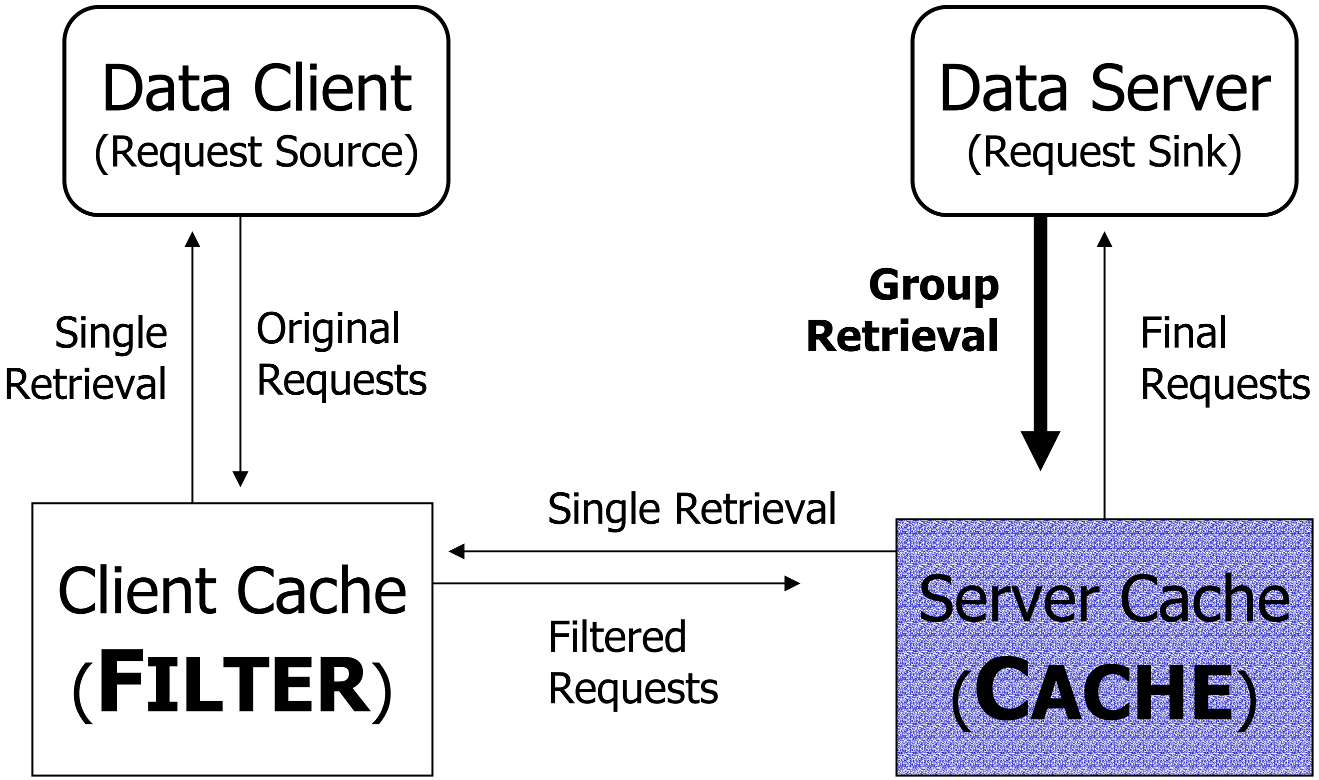
Single-Stage Client Caching
(original workload observed at the client)



Server-Side Caching
(filtered workload observed at the server)

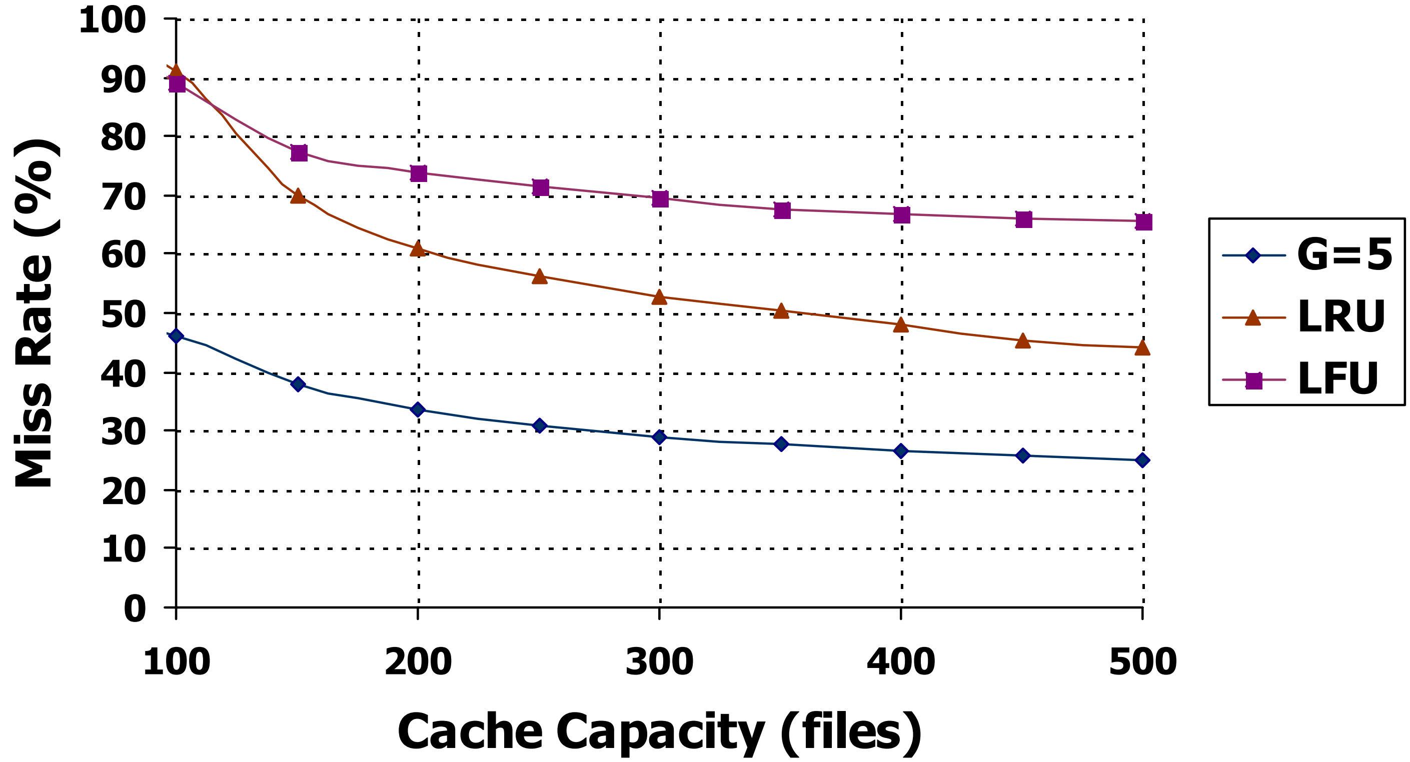


Aggregating Cache
(used for server-side caching)



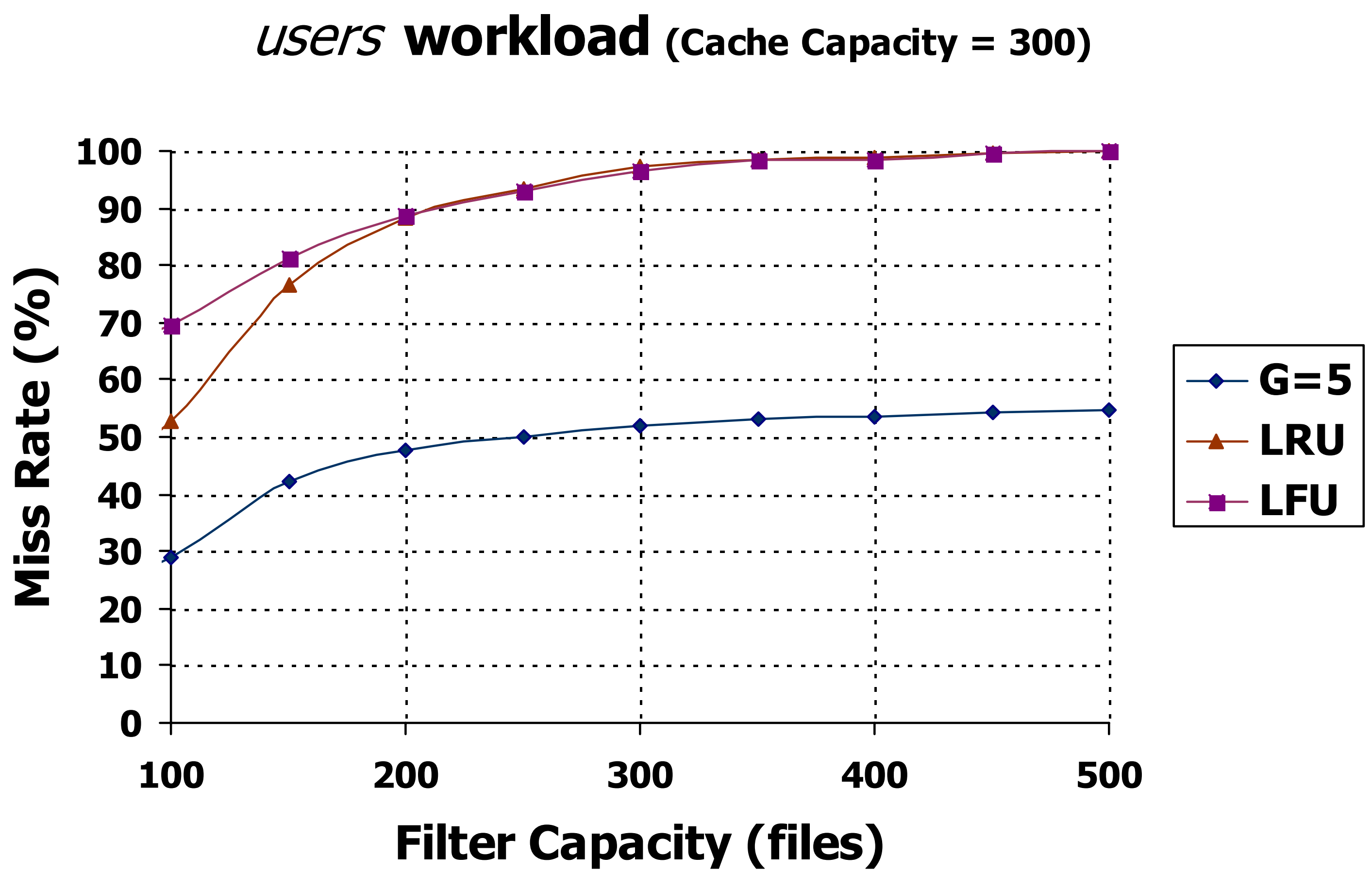
Aggregating Cache
Miss Rates (with client cache filtering)

users **workload** **(Filter Capacity = 100)**



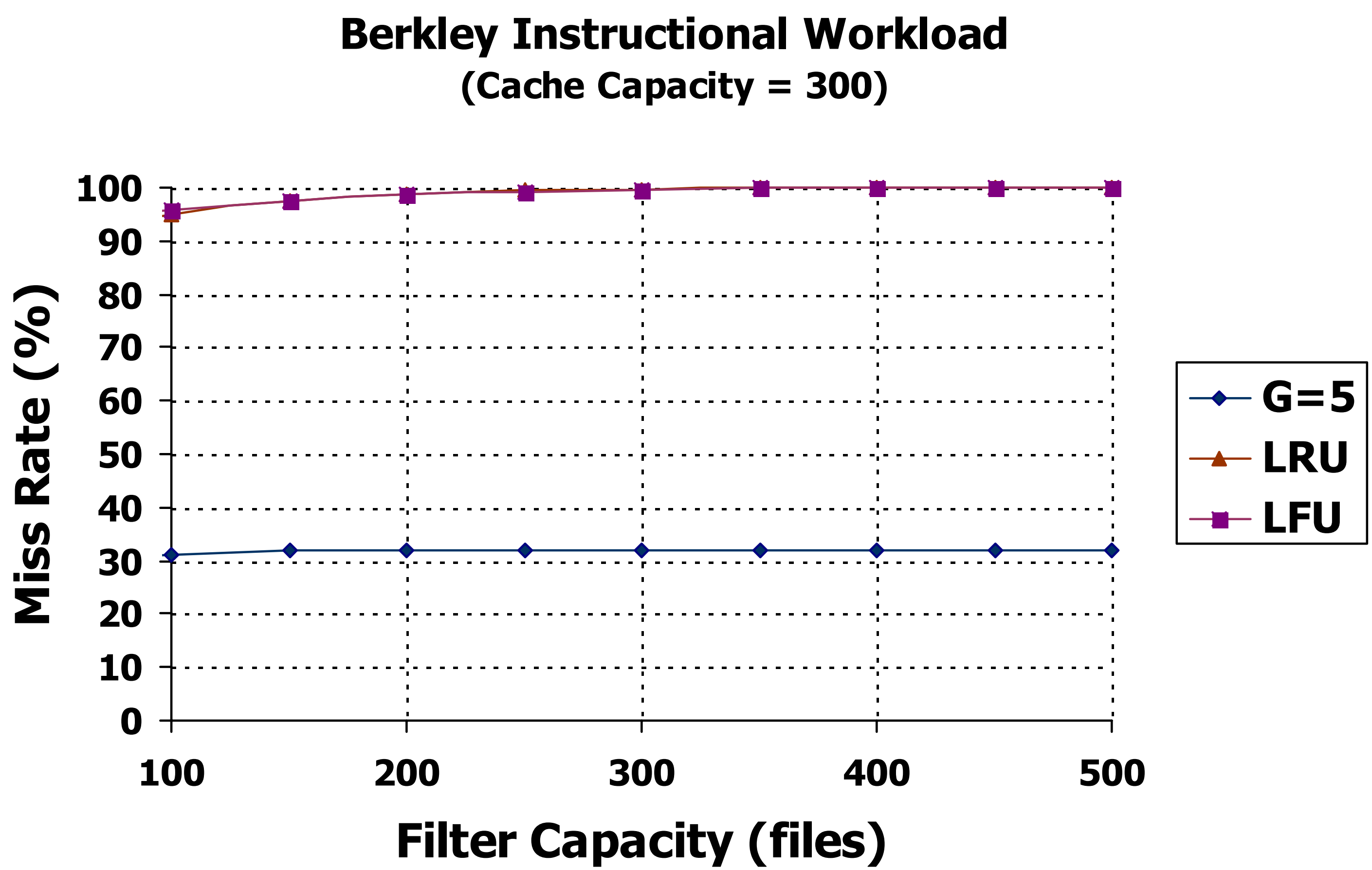
Aggregating Cache

Miss Rates (with client cache filtering)



SSRC

Aggregating Cache
Miss Rates (with client cache filtering)



Visualizing Caching Effects

- Why do aggregating caches still work?
  - Intervening caches do not reduce access predictability
- How can we demonstrate this?
  - Using a new visualization tool (developed in collaboration with the UCSC Viz group) we produce **Cache-Frequency Plots**
  - These are based on **successor entropy**, a single context-based predictability measure

Successor Entropy

- Traditional Self-Information (Entropy)
  - Higher values imply greater unpredictability
  - Predictability of an independent sequence
  - No context information
- Successor Entropy
  - Entropy of individual successor sequences calculated for each file accessed
  - Presented as a **Predictability Histogram**

SSRC

Successor Entropy

- Traditional Self-Information (Entropy)
  - weighted sum of **independent** log-likelihoods

$$H = -\sum_i P(s_i) \cdot \log(P(s_i))$$

- Conditional entropy
  - given knowledge that condition *c* is true

$$H(c) = -\sum_i P(s_i \mid c) \cdot \log(P(s_i \mid c))$$

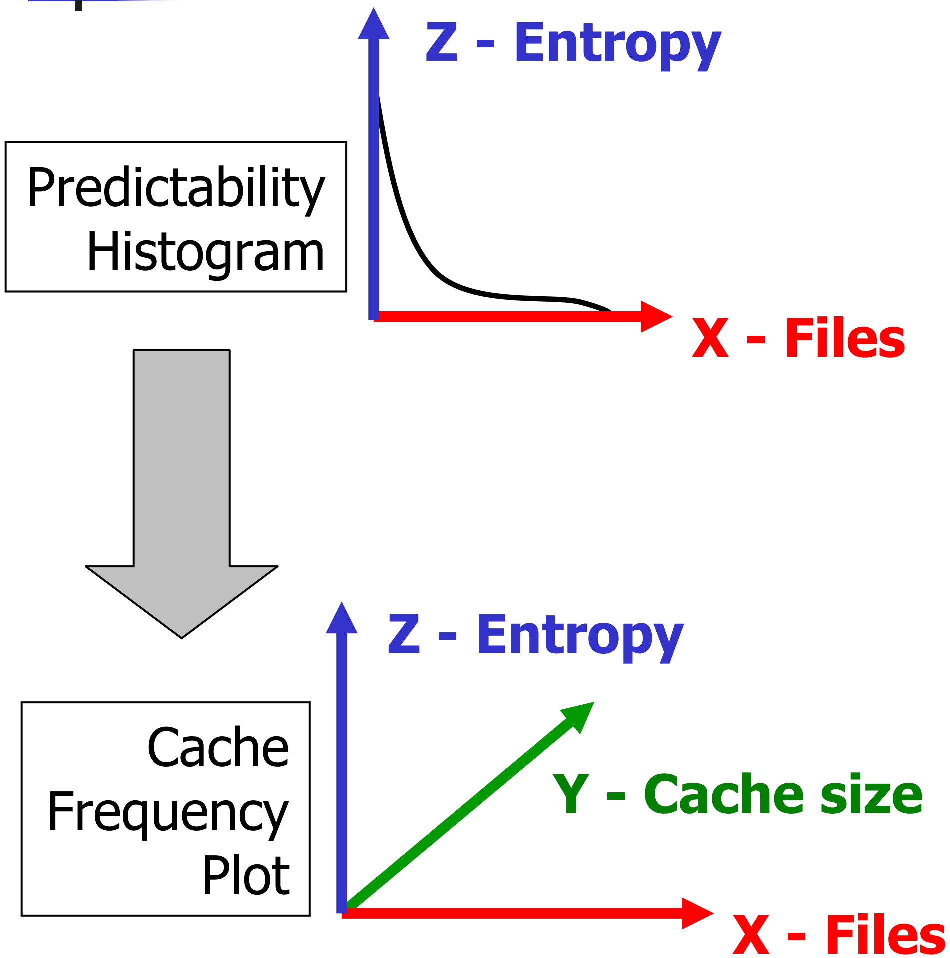
Successor Entropy

- Given observed accesses to ***m*** successors ***s_i*** of file **a**, we define the successor entropy of file **a** as:

$$H(a) = -\sum_{i=1}^{m} P(s_i \mid a) \cdot \log(P(s_i \mid a))$$

SSRC

Cache-Frequency Plots

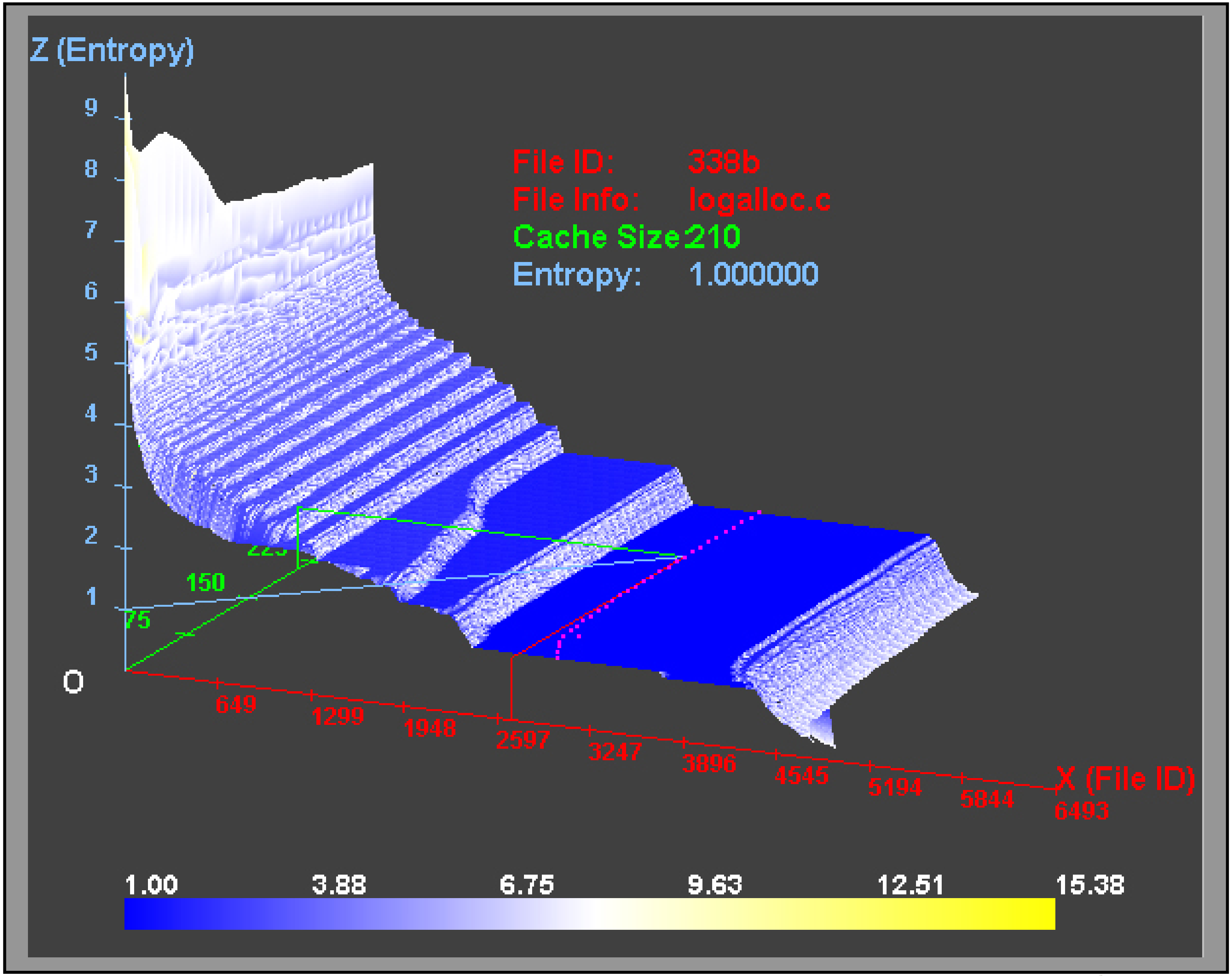


- X-axis
  - Files, ordered by decreasing Z-value
- Y-axis
  - Filtering cache sizes
- Z-axis
  - Successor entropy
- Surface-Point Color
  - File access frequency

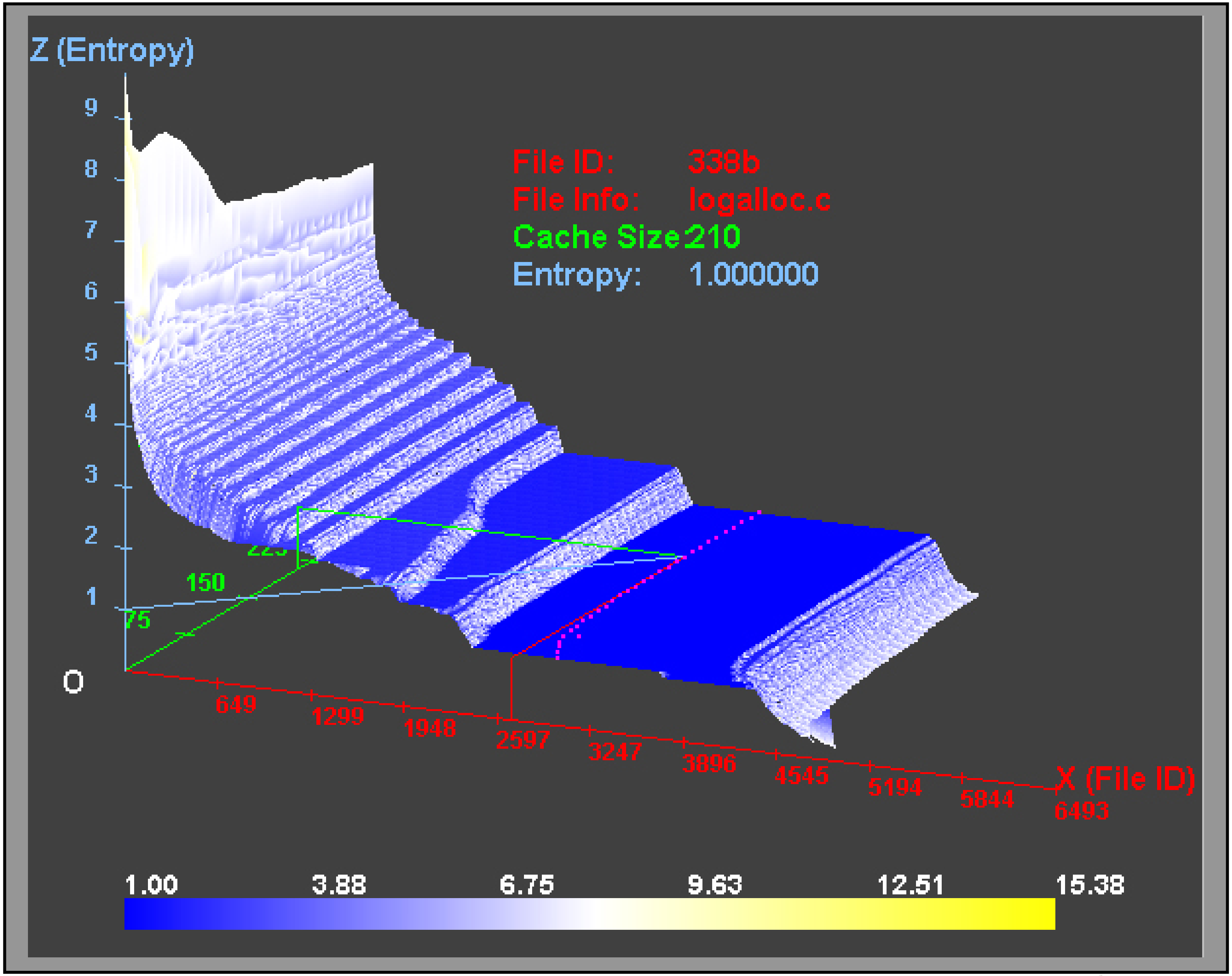
Cache-Frequency Plots (cont'd)

- Predictability histogram
  - Demonstrates variation in file access predictability
- The Cache-Frequency Plots
  - Effects of intervening cache sizes on predictability histograms
  - Correlation between file popularity (access frequency) and successor predictability

SSRC



Z (Entropy)
9
8
7
6
5
4
3
2
1
O
75
150
File ID: 338b
File Info: logalloc.c
Cache Size 210
Entropy: 1.000000
649
1299
1948
2597
3247
3896
4545
5194
5844
6493
X (File ID)
1.00
3.88
6.75
9.63
12.51
15.38


Z (Entropy)
9
8
7
6
5
4
3
2
1
O
75
150
File ID: 338b
File Info: logalloc.c
Cache Size 210
Entropy: 1.000000
649
1299
1948
2597
3247
3896
4545
5194
5844
6493
X (File ID)
1.00
3.88
6.75
9.63
12.51
15.38

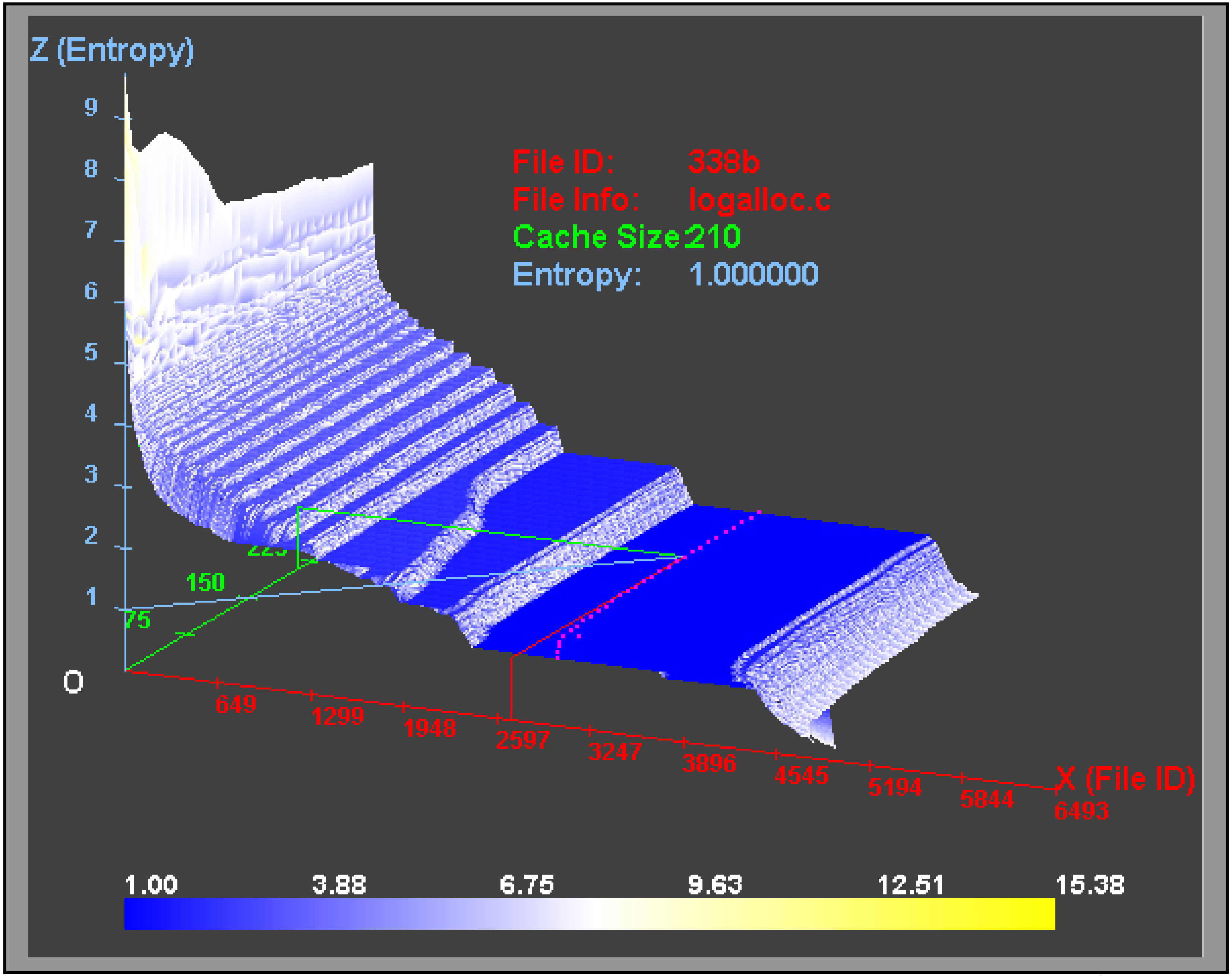
SSRC

Predictability Results

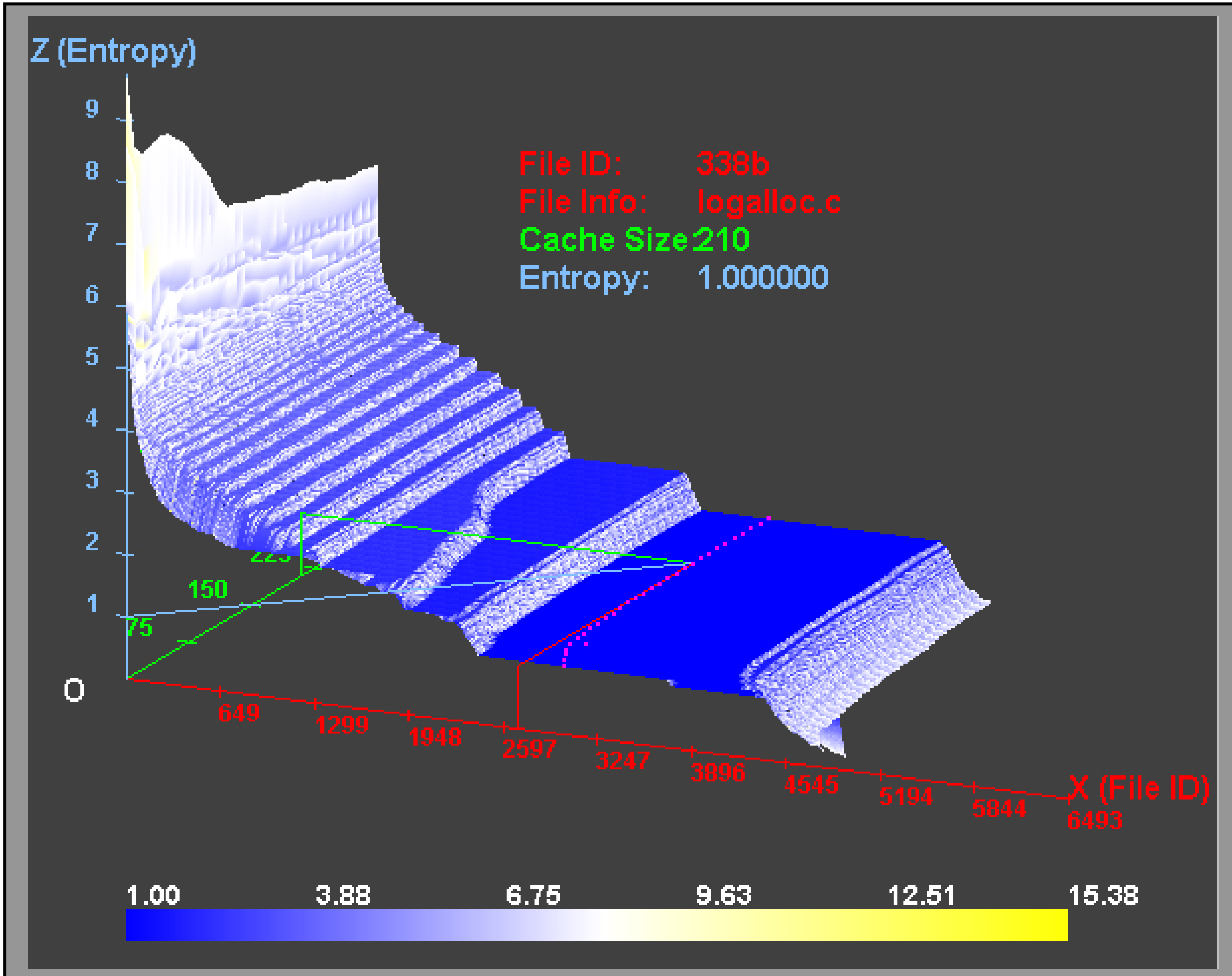
- File successor predictability varies as dramatically as file popularity
  - High skew among file successor entropy
  - Most have highly predictable successors
- Predictability independent of popularity
  - Some of the most popular files have the most predictable successor behavior

Caching Effects

- Increasing the capacity of intervening caches …
  - … reduces the skew of access frequencies, by reducing the number of very high-frequency and unpredictable files
  - … actually increases predictability, and reduces the variation among files


Z (Entropy)
9
8
7
6
5
4
3
2
1
O
75
150
225
File ID: 338b
File Info: logalloc.c
Cache Size 210
Entropy: 1.000000
649
1299
1948
2597
3247
3896
4545
5194
5844
6493
X (File ID)
1.00
3.88
6.75
9.63
12.51
15.38

SSRC


Z (Entropy)
9
8
7
6
5
4
3
2
1
O
150
75
File ID: 338b
File Info: logalloc.c
Cache Size 210
Entropy: 1.000000
649
1299
1948
2597
3247
3896
4545
5194
5844
6493
X (File ID)
1.00
3.88
6.75
9.63
12.51
15.38

Related Work

- File Access Prediction
  - Krishnan, Griffioen, Duchamp, and Kroeger
- Mobile File Hoarding
  - Coda, and SEER
- Web Caching
  - Bestavros, Duchamp, and Wolman

Conclusions

- Aggregating cache
  - Most files see few unique successors
  - Simple grouping can significantly reduce demand cache misses while providing implicit pre-fetching
  - Can maintain reasonable hit rates in the presence of cache filtering effects

Conclusions (cont'd)

- No pre-fetch timing issues
  - Explicit pre-fetching may hurt performance, and demands timeliness
  - Relationship tracking is an optional activity that can be safely delayed/ignored
- If you have a client cache and a server cache, you want to do this!

SSRC

Ongoing and Future Work

- Examine alternate predictors
  - Program-based predictors (Yeh *et al.*)
- Partial file transfer, block-level grouping
- Storage allocation & placement problems
- Mobile applications
- Multi-level caches

Further Information & Questions?

http://ssrc.cse.ucsc.edu/

http://hssl.cs.jhu.edu/

darrell@cs.ucsc.edu

a.amer@acm.org

randal@cs.jhu.edu

SSRC