Using Multiple Predictors to Improve the Accuracy of File Access Predictions

Gary A. S. Whittle, U of Houston
Jehan-François Pâris, U of Houston
Ahmed Amer, U of Pittsburgh
Darrell D. E. Long, UC Santa Cruz
Randal Burns, Johns Hopkins U

THE PROBLEM

- ***Disk drive capacities*** double every year
  - Better than the 60% per year growth rate of ***semiconductor memories***
- ***Access times*** have decreased by a factor of 3 over the last 25 years
- ***Cannot keep up*** with increased I/O traffic resulting from faster CPUs
- Problem is ***likely to become worse***

Possible Solutions (I)

- ***"Gap filling" technologies***
  - Bubble memories (70's)
  - Micro electro-mechanical systems (MEMS)
  - These devices must be at the same time
    - Much faster than disk drives
    - Much cheaper than main memory
  - Hard to predict which technology will win

Possible Solutions (II)

- ***Software Solutions***
  - Aim at masking disk access delays
  - Long successful history
  - Two main techniques
    - ***Caching***
    - ***Prefetching***

Caching

- Keeps in memory recently accessed data
- Used by nearly all systems
- Scale boosted by availability of cheaper RAM
  - Should cache entire small files
- Small penalty for keeping in a cache data that will not be reused
  - Only reduces cache effectiveness

Prefetching

- Anticipates user needs by loading into cache data before they are needed
- Made more attractive by availability of cheaper RAM
- Hefty penalty for bringing into main memory data that will not be used
  - Results in additional I/O traffic
- Most systems err on the side of caution

OUR APPROACH

- We want to improve the performance of prefetching by improving the accuracy of our file access predictions
- We need better ***file access predictors***
- These better predictors could be used
  - To reduce the number of incorrect prefetches
  - To group together on disk data that are needed at the same time

Our Criteria

- A good file predictor should
  - Have reasonable space and time requirements
    - *Cannot keep a long file access history*
  - Make as many successful predictions as possible
  - Make as few bad predictions as feasible

PREVIOUS WORK

- Two major approaches:
  - ***Complex predictors***
  - ***Very simple predictors***

Complex Predictors

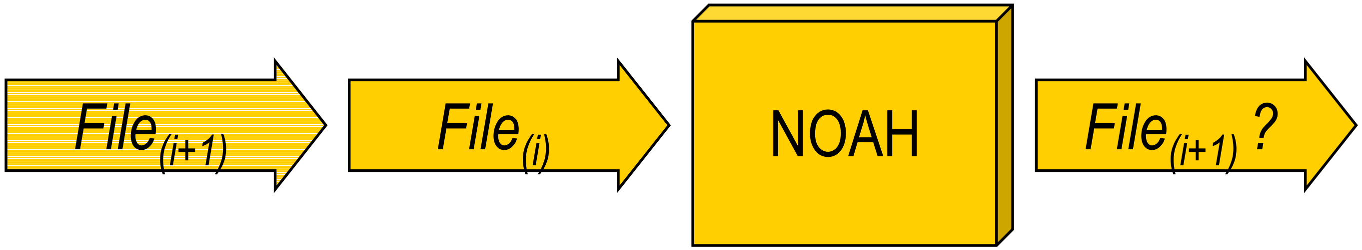
- Collect data from a long file access history and store them in a compressed form
  - *Fido* (Palmer *et al*., 1991)
  - Graph-based relationships (Griffioen and Appleton, 1994)
  - Detecting file access patterns (Tait *et al*., 1991 and Lei and Duchamp, 1997)
  - Context modeling and data compression (Kroeger and Long, 2001)

Simple Predictors

- ***Last Successor***:
  - If file ***B*** was preceded by file ***A*** the last time ***B*** was accessed, predict that ***B*** will will be the successor of ***A***
    (Lei and Duchamp, 1997)
- ***Stable Successor*** (Amer and Long, 2001)
- ***Recent Popularity*** (Amer *et al.*, 2002)

Stable Successor (Noah)

- Maintains a current prediction for the successor of every file
- Changes current prediction to last successor if last successor was repeated for ***S*** subsequent accesses
  - ***S*** (stability) is a parameter, default = 1



Example

- Assume sequence of file accesses

  A B C E A B A F D A G A G A ?

  and ***S*** = 1
- Stable successor will predict ***B*** as the successor of ***A*** and not update this prediction until it has observed two consecutive instances of ***G*** following ***A***

Recent Popularity

- Also known as **Best *j-out-of-k***
- Maintains a list of the ***k*** most recently observed successors of each file
- Searches for the most popular successor from the list
- Predict that file if it occurred at least ***j*** times in the list
- Uses *recency* to break possible ties

OUR PREDICTOR

- Combines several simple heuristics
- Can include *specialized heuristics* that
  - Can make very accurate predictions
  - But only in some specific case
- More accurate predictions
- No significant additional overhead
  - All our predictors base their prediction on the same data

Performance Criteria (I)

- Two traditional metrics
  - *success-per-reference*
  - *success-per-prediction*
- Neither of them is satisfactory
  - *success-per-reference* favors heuristics that always make a prediction
  - *success-per-prediction* favors heuristics that are exceedingly cautious

Performance Criteria (II)

- Our new performance criterion: ***effective-miss-ratio***

$$\frac{N_{ref} - N_{corr} + \alpha N_{incorr}}{N_{ref}}$$

where $0 \leq \alpha \leq 1$ is a coefficient representing the cost of an incorrect prediction

Performance Criteria (III)

- $\alpha = 0$ means that we can always preempt the fetch of a file that was incorrectly predicted
- $\alpha = 1$ means that we can never do that

Experimental Setup

- We selected four basic heuristics and simulated their application to two sets of traces
  - Four traces collected at CMU: *mozart, ives, dvorak* and *barber*
  - Three traces collected at UC Berkeley: *instruct, research* and *web*

The Four Base Heuristics

- Most Recent Consecutive Successor
- Predecessor Position
- Pre-Predecessor Position
- *j*-out-of-*k* Ratio for Most Frequent Successor

Most Recent Consecutive Successor

- If we encounter the file reference sequence

  A B C B C B C B ?

  we predict ***C***
- S*uccess-per-prediction* increases linearly as the number of consecutive successors increases from one through three
- More than six most recent consecutive successors are a strong indicator that this successor will be referenced next

Predecessor Position

- If the file reference sequence ***<u>ABC</u>*** occurred in the recent past, we predict ***C*** whenever the sequence ***<u>AB</u>*** is present
- Can yield prediction accuracies between 55 and 90 percent

Pre-Predecessor Position

- Extension of previous heuristics
- If the file reference sequence ***ABCD*** occurred in the recent past, we predict D when the sequence ***ABC*** reappears
- Can yield prediction accuracies between 65 percent and 95 percent.

j-out-of-k Ratio for Most Frequent Successor

- Similar to Recent Popularity
- Mostly used when none of the previous predictors works

Combining the Four Heuristics

- Assign ***empirical weights*** to the four heuristics
  - Weights are fairly independent of specific access patterns
  - Can use the Berkeley trace to compute weights and use any of the CMU traces in our simulation and *vice versa*
- Empirical weights are used to select the most trustworthy prediction

Avoiding False Predictions (I)

- Our composite predictor includes a ***probability threshold*** whose purpose is to reduce the number of bad predictions
- Only used when $\alpha > 0$
- Threshold increases with value of α and reaches 0.5 when $\alpha = 1$

Avoiding False Predictions (II)

- We added to our predictor a ***confidence measure***
  - 0.0 to 1.0 saturating counter
  - Maintained for each file
  - Initialized to 0.5
  - Incremented by 0.1 after a successful prediction
  - Decremented by 0.05 after an incorrect prediction.

Avoiding False Predictions (III)

- We decline to make a prediction whenever

  confidence measure < threshold

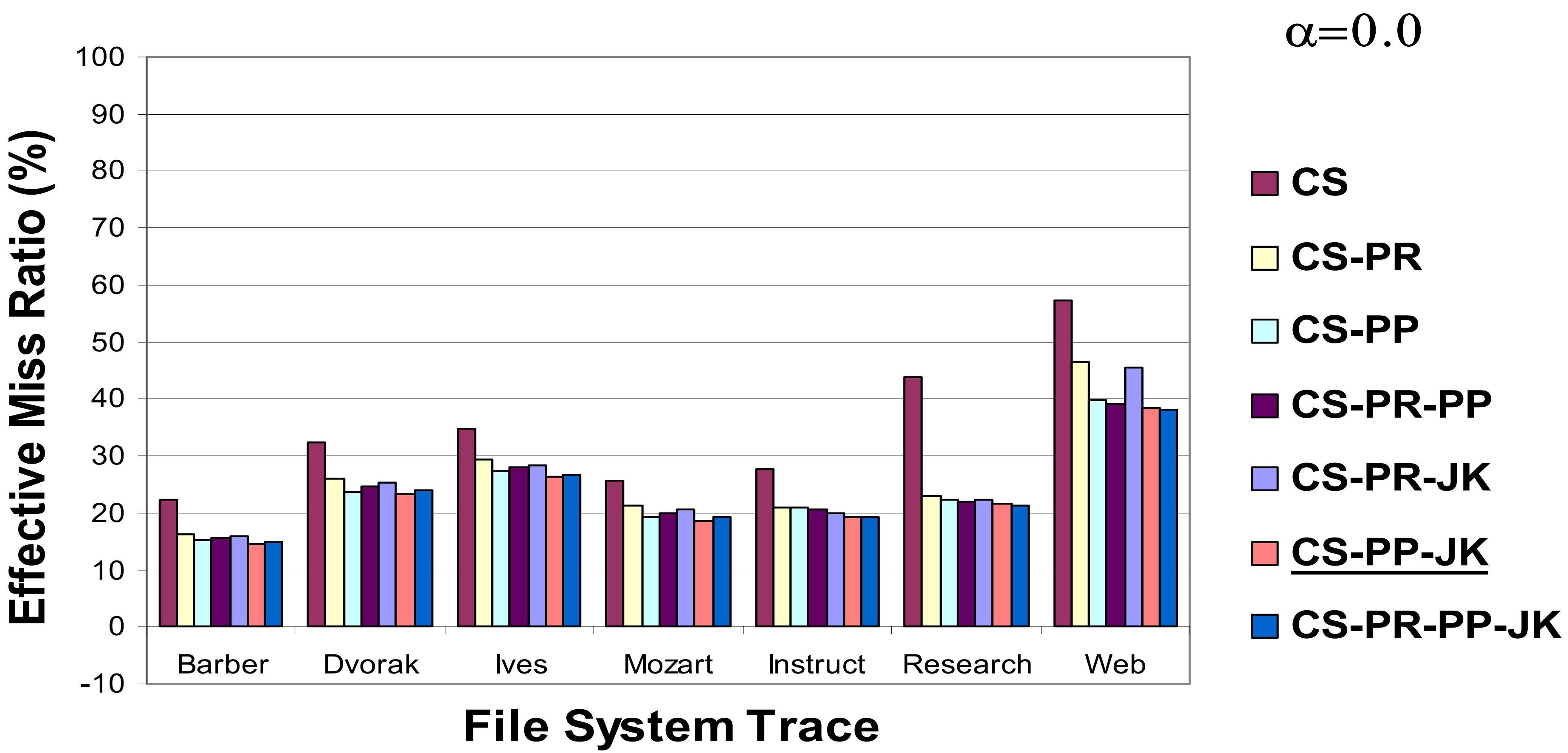
Cost Reduction

- We compared using
  - A successor history length of 9 file identifiers
  - A successor history length of 20 file identifiers
- *Effective-miss-ratios* were within 1% of each other
- Can safely reduce length of successor history to 9 file identifiers per file

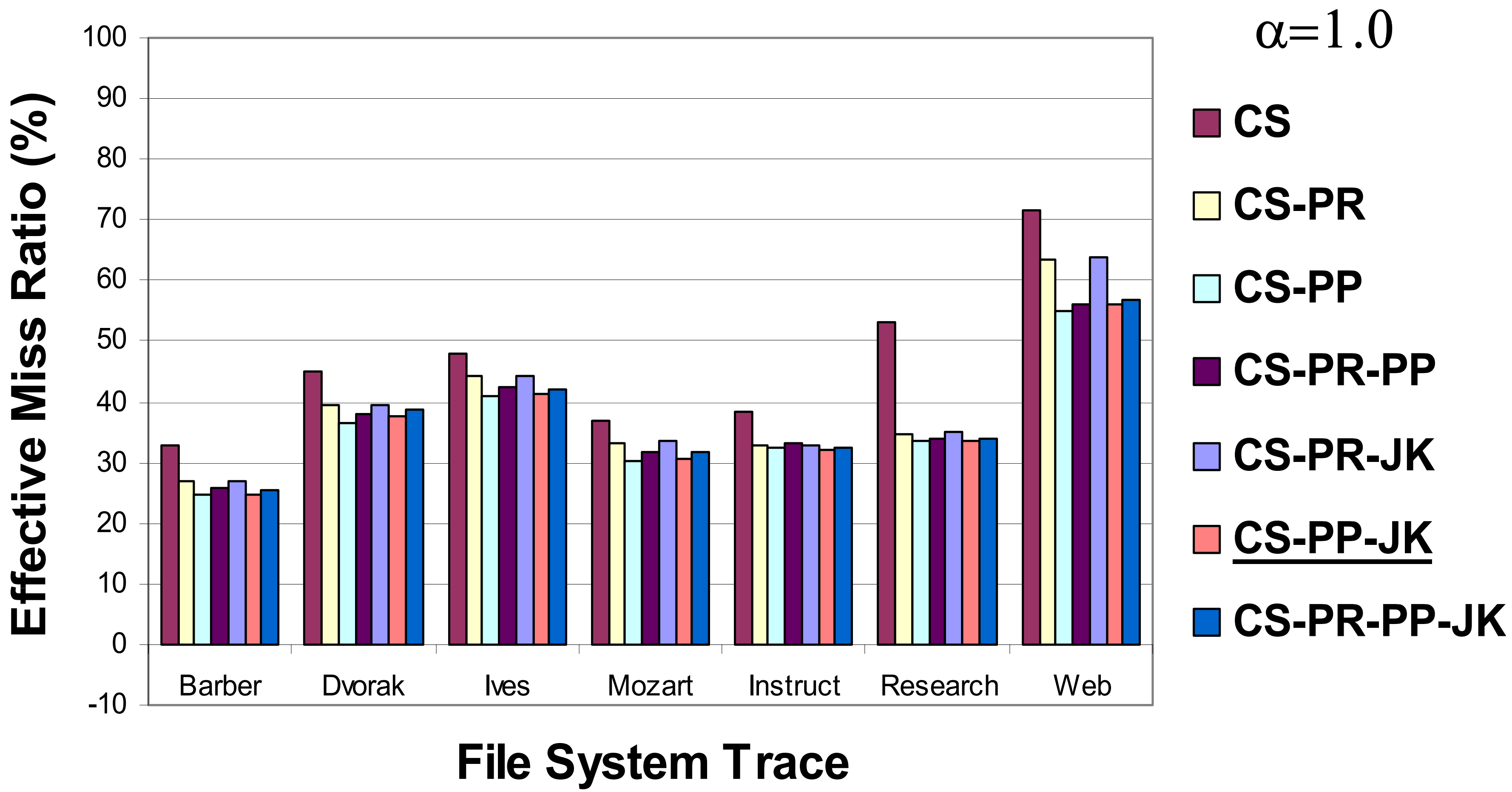
EXPERIMENTAL RESULTS

- Our composite predictor used
  - All four heuristics
  - Mean heuristic weights
  - A successor history length of 9 file identifiers
  - A confidence measure
- Results for the *First-Successor* predictor were not included
  - Much worse than all other predictors

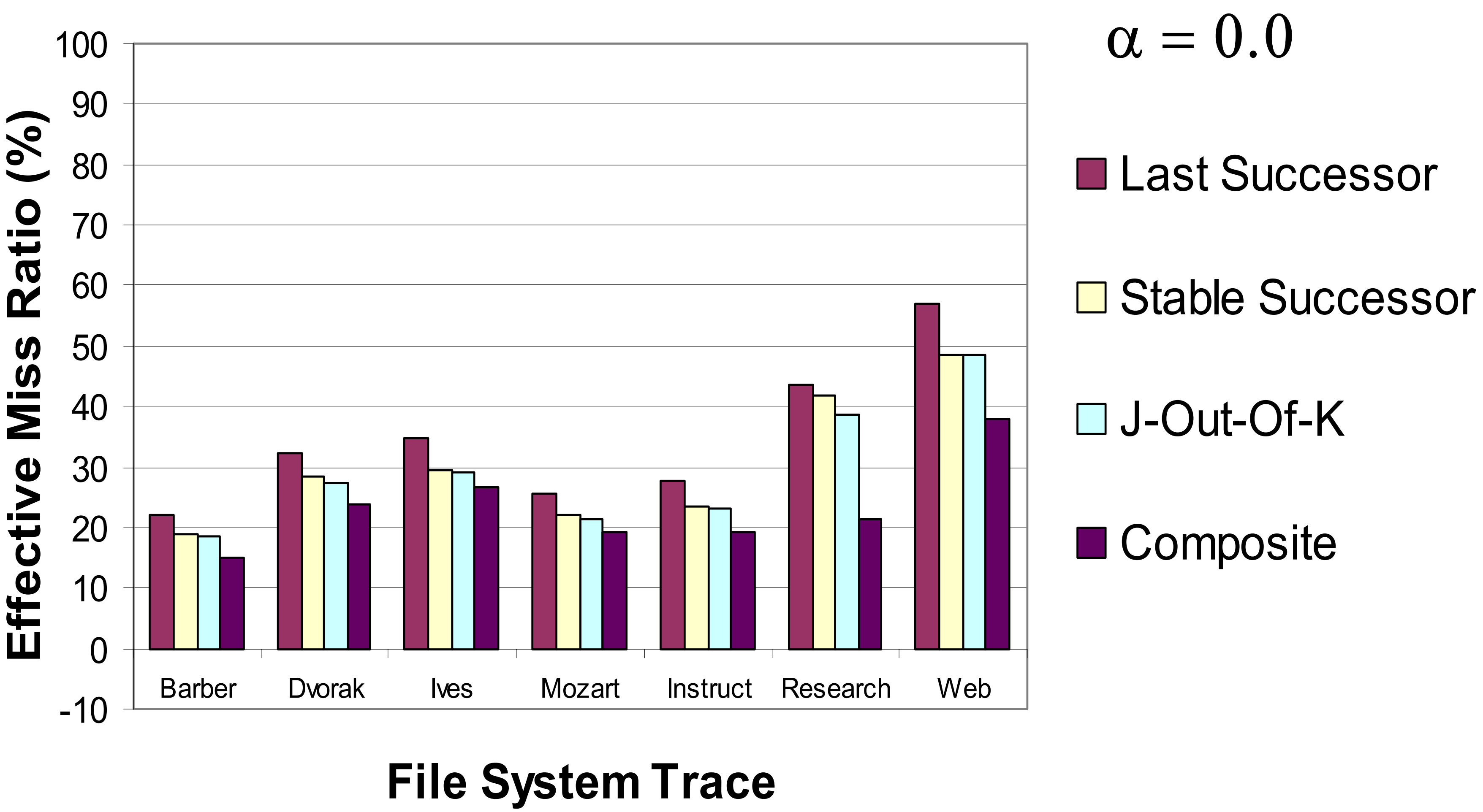
Comparing the Heuristics (I)



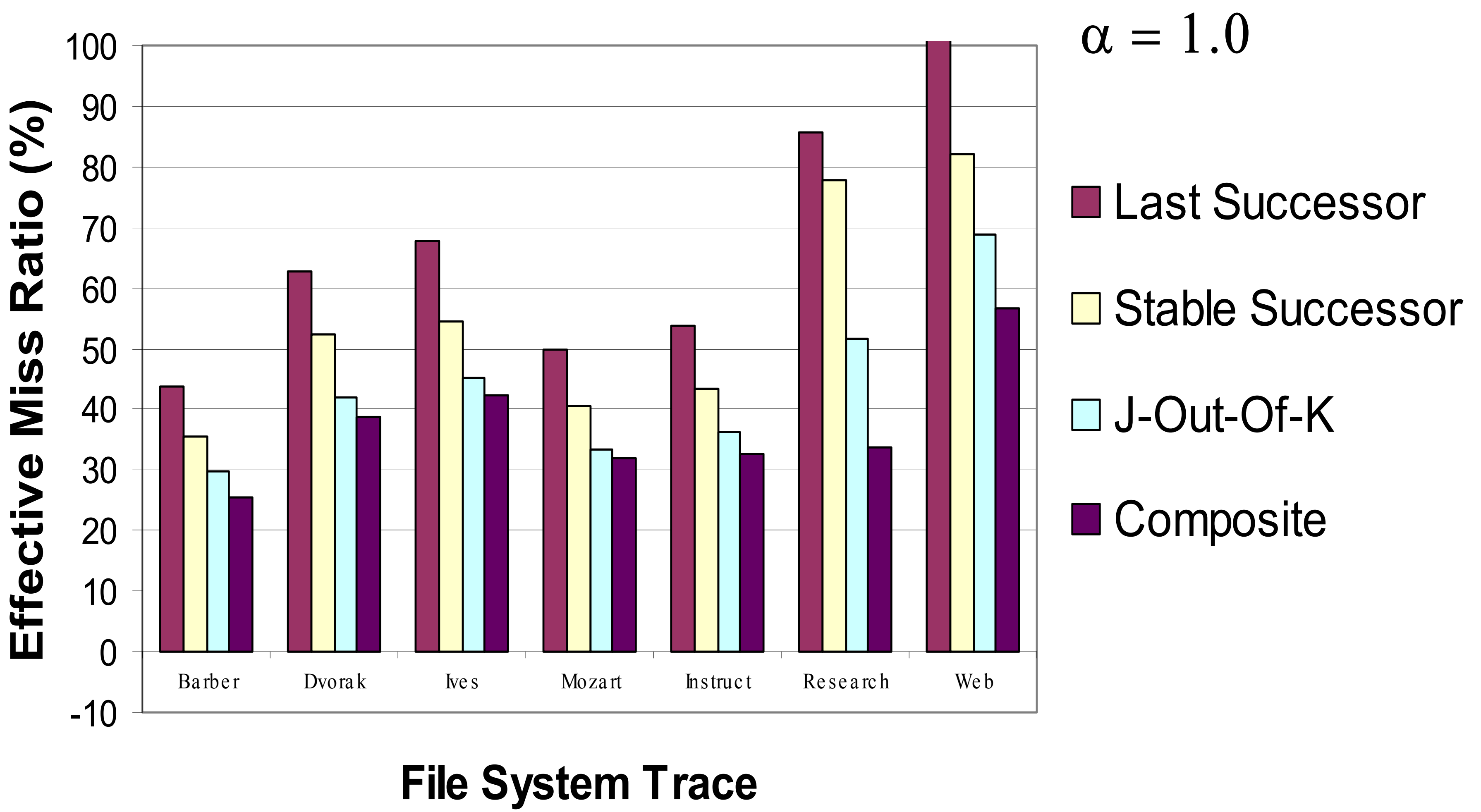
Comparing the Heuristics (II)



Overall Performance (I)



Overall Performance (II)



CONCLUSIONS

- Our composite predictor provides lower effective miss ratios than other simple predictors
- More work is needed
  - Find better ways to evaluate the predictions of the four heuristics
  - Eliminate redundant heuristics: Predecessor Position is a good candidate