

# Generating Realistic Impressions for File-System Benchmarking

**Nitin Agrawal**

Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau

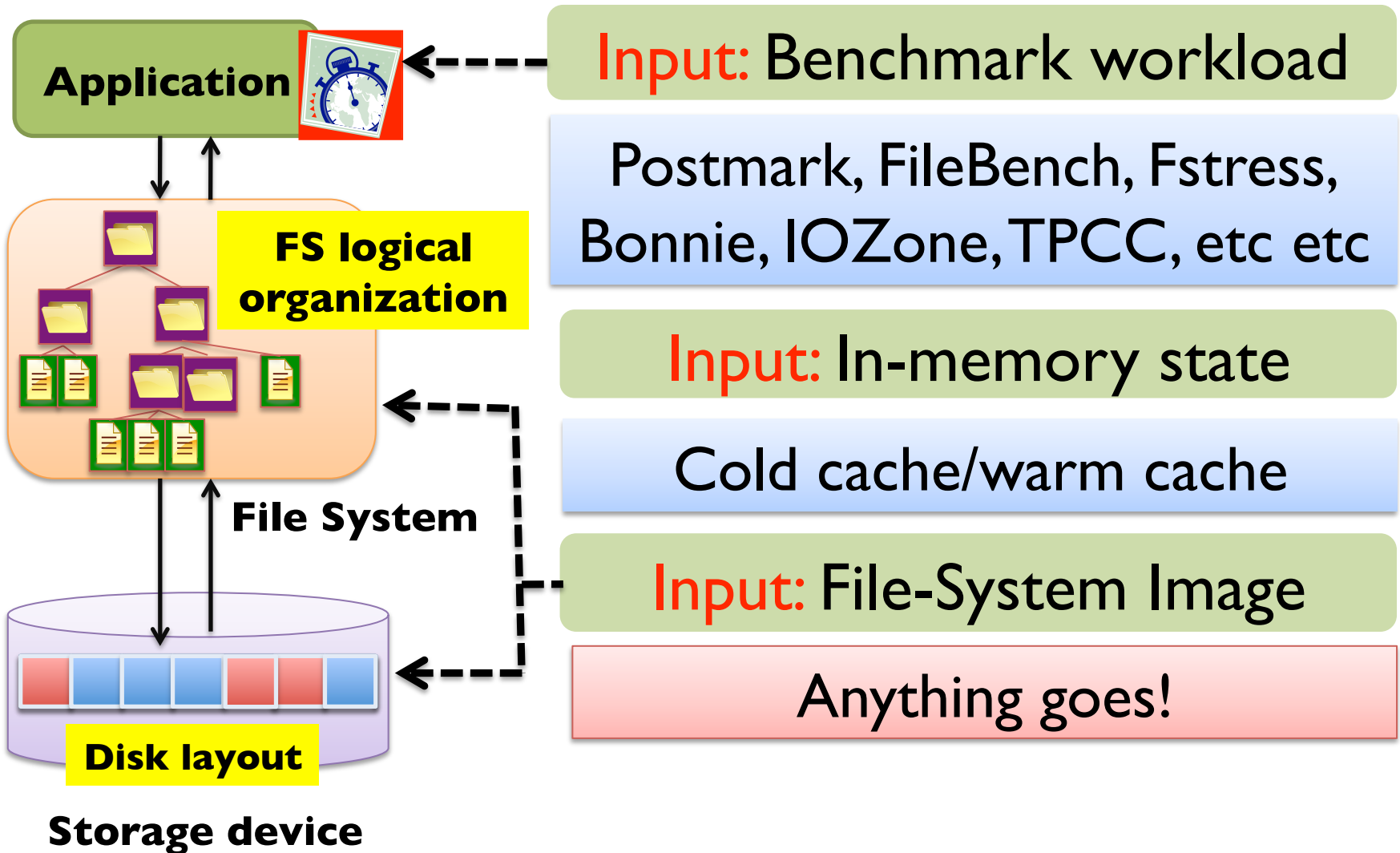


THE UNIVERSITY  
*of*  
**WISCONSIN**  
MADISON

**“For better or for worse,  
benchmarks shape a field”**

*David Patterson*

# Inputs to file-system benchmarking



# FS images in past: use what is convenient

**Typical** desktop file system w/ no description (SOSP 05)

5-deep tree, 5 subdirs, 10 8KB files in each (FAST 04)

Randomly generated files of **several** MB (FAST 08)

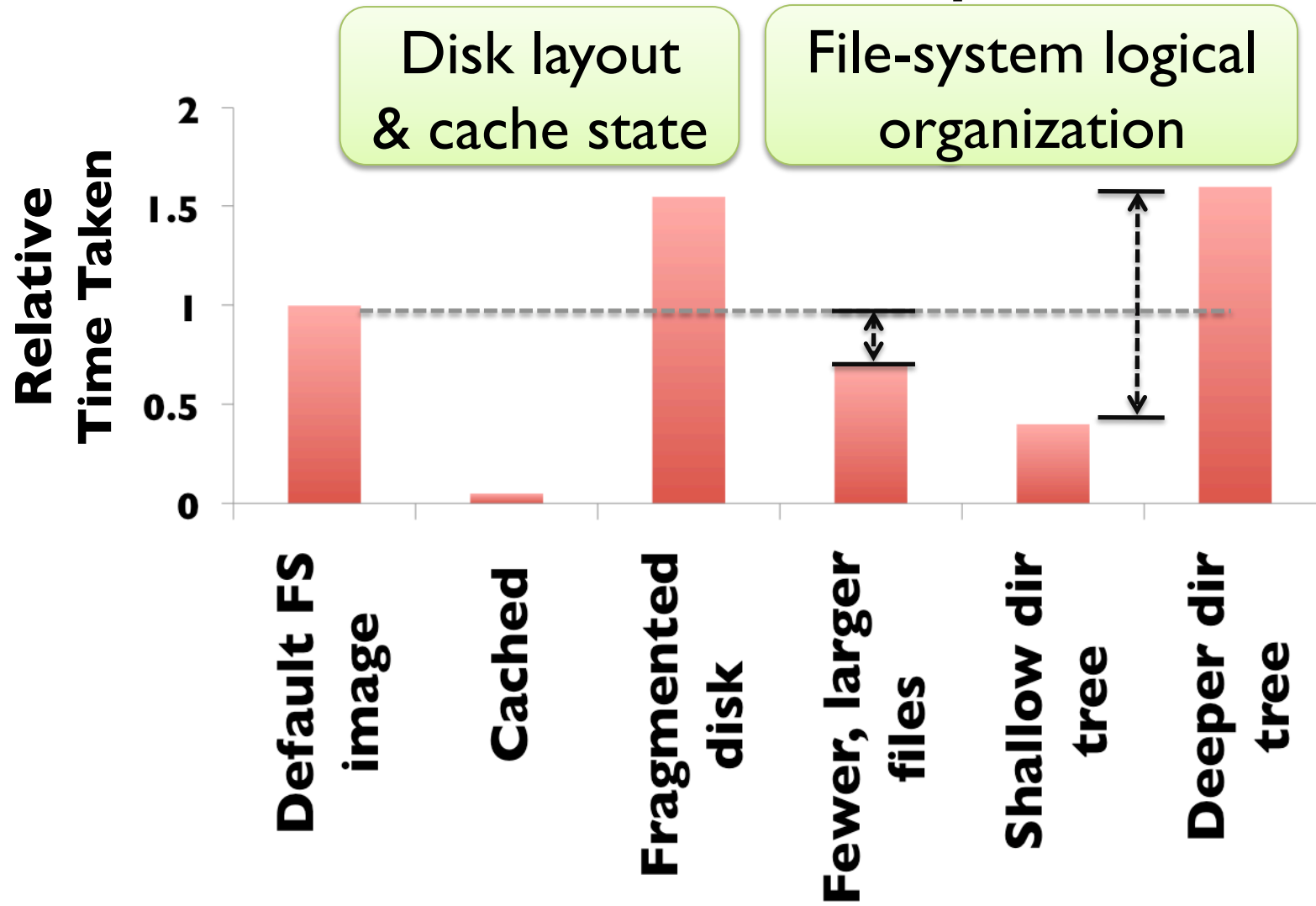
1000 files in 10 dirs w/ **random** data (SOSP 03)

188GB and 129GB volumes in **Engg** dept (OSDI 99)

10702 files from **/usr/local**, size 354MB (SOSP 01)

1641 files, 109 dirs, 13.4 MB total size (OSDI 02)

# Performance of *find* operation



# Problem scope

Characteristics of file-system images  
have strong impact on performance

We need to incorporate representative  
file-system images in benchmarking & design

How to create **representative**  
file-system images?

# Requirements for creating FS images

- Access to data on file systems and disk layout
  - Properties of file-system metadata [Satyanarayan81, Mullender84, Irlam93, Sienknecht94, Douceur99, Agrawal07]
  - Disk fragmentation [Smith97]
  - More such studies in future?
- A technique to create file-system images that is
  - **Representative**: given a set of input distributions
  - **Controllable**: supply additional user constraints
  - **Reproducible**: control & report internal parameters
  - **Easy to use**: for widespread adoption and consensus

# Introducing *Impressions*

- Powerful statistical framework to generate file-system images
  - Takes properties of file-system attributes as input
  - Works out underlying statistical details of the image
  - Mounted on a disk partition for real benchmarking
  - Satisfies the four design goals
- Applying Impressions gives useful insights
  - What is the impact on performance and storage size?
  - How does an application behave on a real FS image?

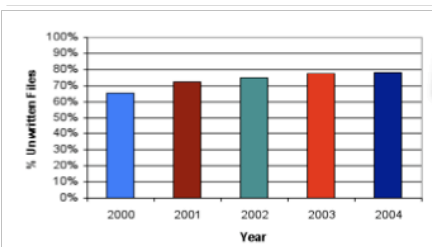
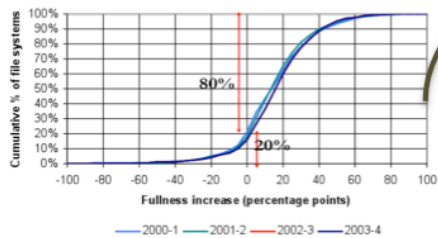
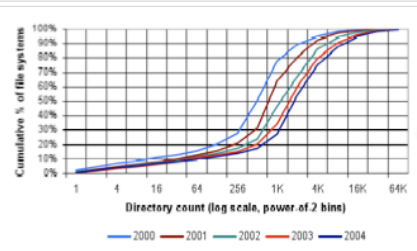
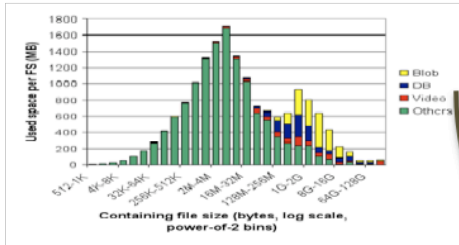


# Outline

- Introduction
- **Generating realistic file-system images**
- Applying Impressions: Desktop search
- Conclusion

# Overview of Impressions

## FS Distributions

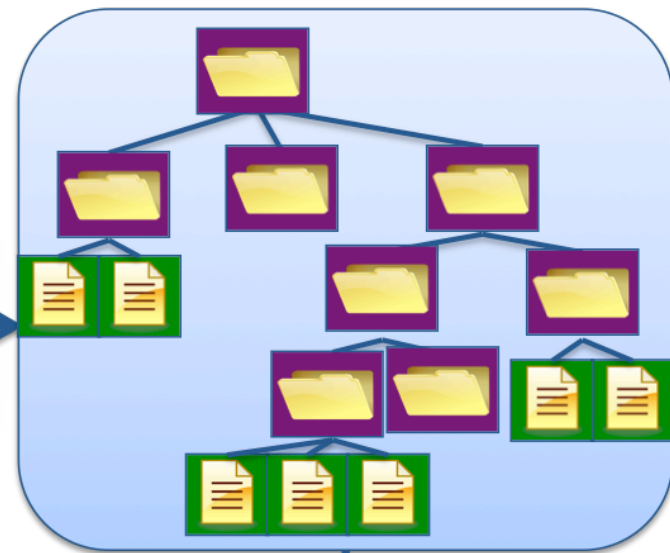


User specified FS parameters



Impressions

File-system state



On-disk state

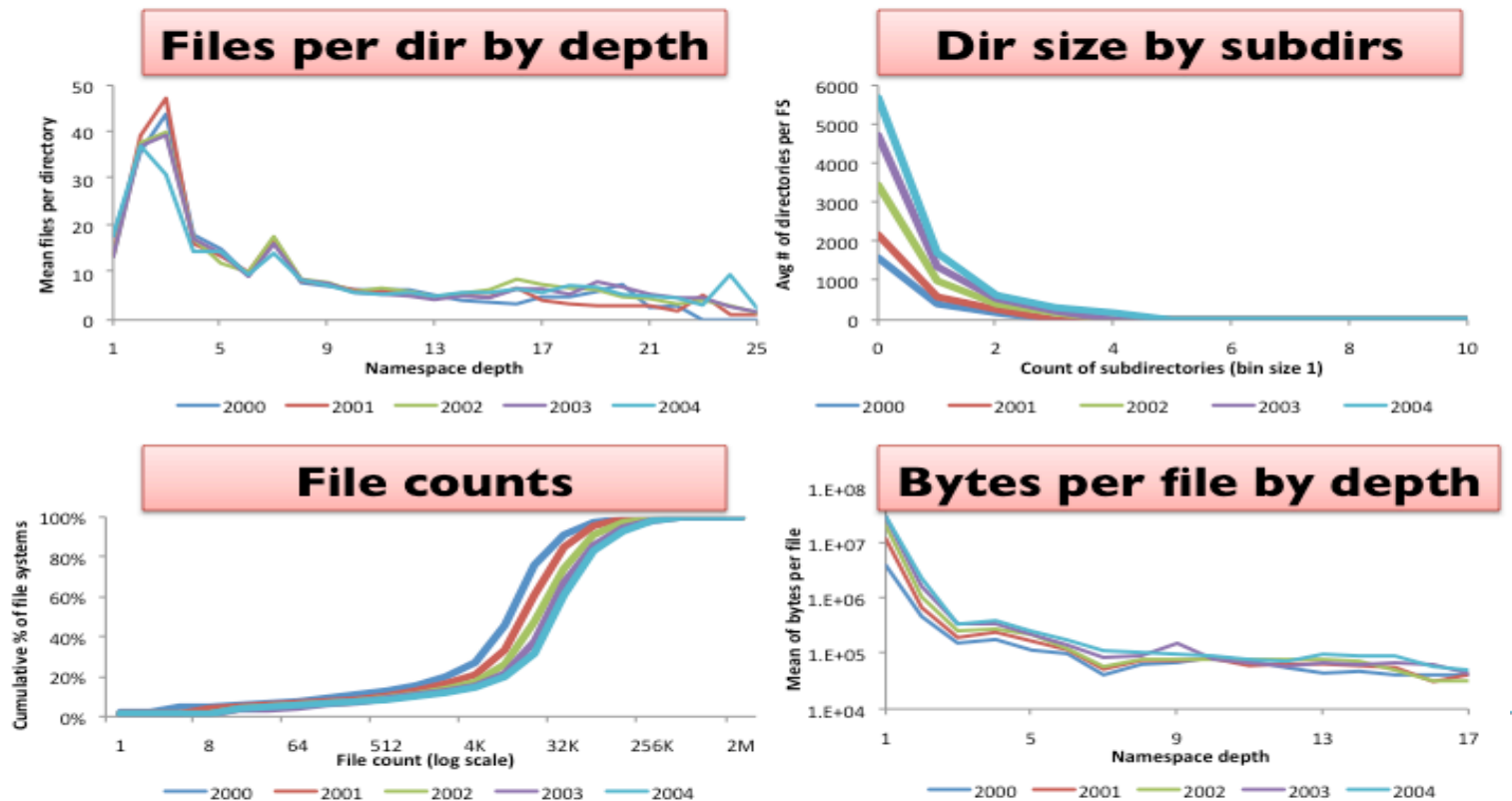


# Properties of file-system metadata

“Five-year study of file-system metadata” [FAST07]

(Agrawal, Bolosky, Douceur, Lorch)

Used as exemplar for metadata properties in Impressions

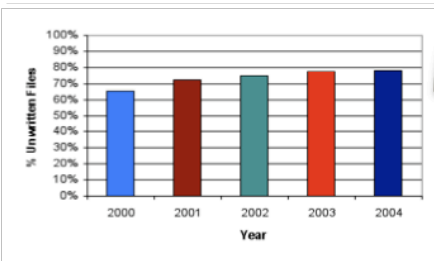
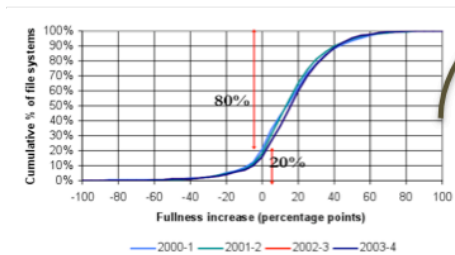
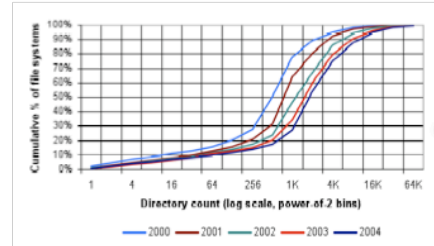
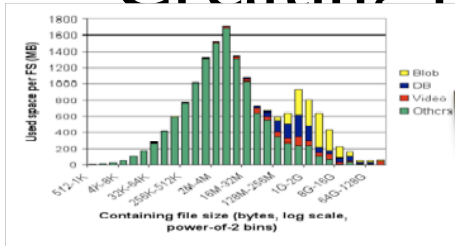


# Features of Impressions

- Modes of operation for different usages
  - Basic mode: choose default settings for parameters
  - Advanced mode: several individually tunable knobs
- Thorough statistical machinery ensures accuracy
  - Uses parameterized curve fits
  - Allows arbitrary user constraints
  - Built-in statistical tests for goodness-of-fit
- Generates namespace, metadata, file content, and disk fragmentation using above techniques

# Creating valid metadata

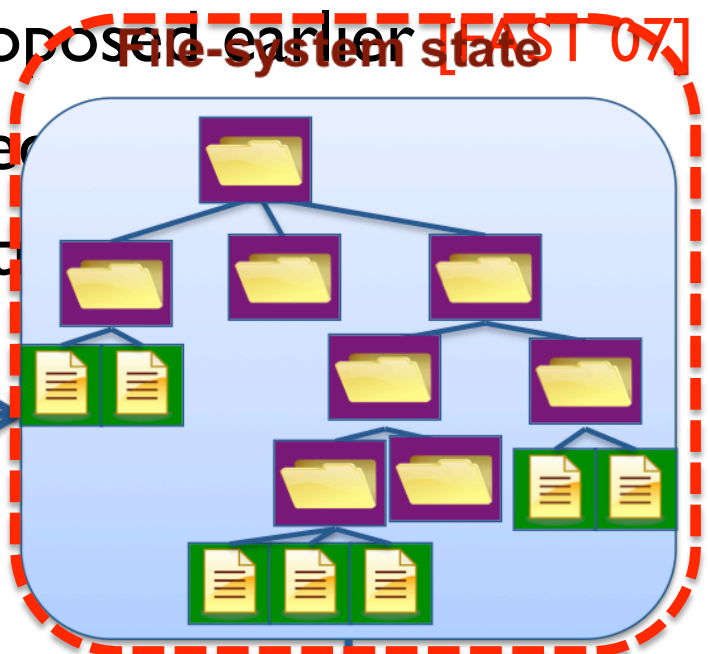
## FS Distributions



User specified FS parameters



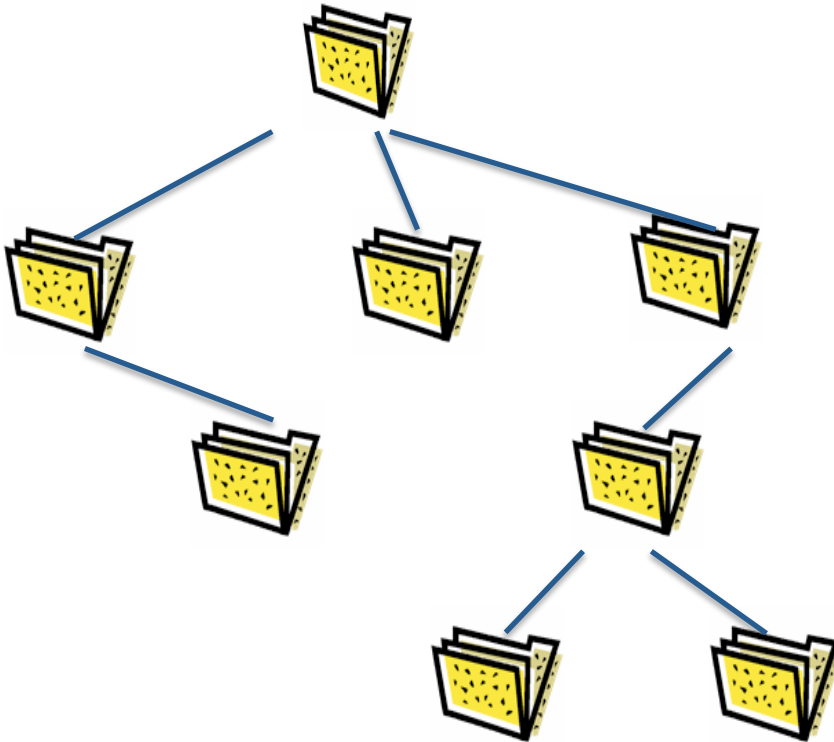
generative Model proposed earlier [FAST 07]  
 the process of directly regenerates distribution of directory depth



On-disk state



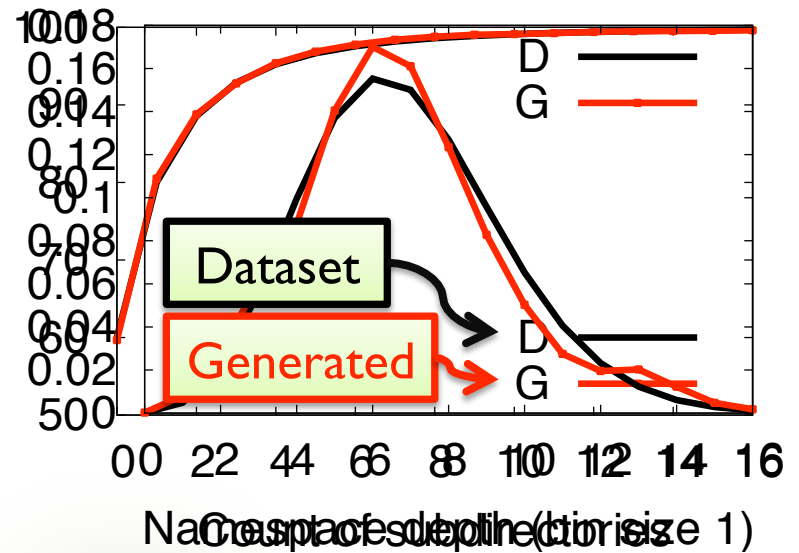
# Creating namespace



Probability of parent selection  
 $\approx \text{Count}(\text{subdirs}) + 2$

## Dirs by namespace depth

Cumulative of directories



**Directory tree**

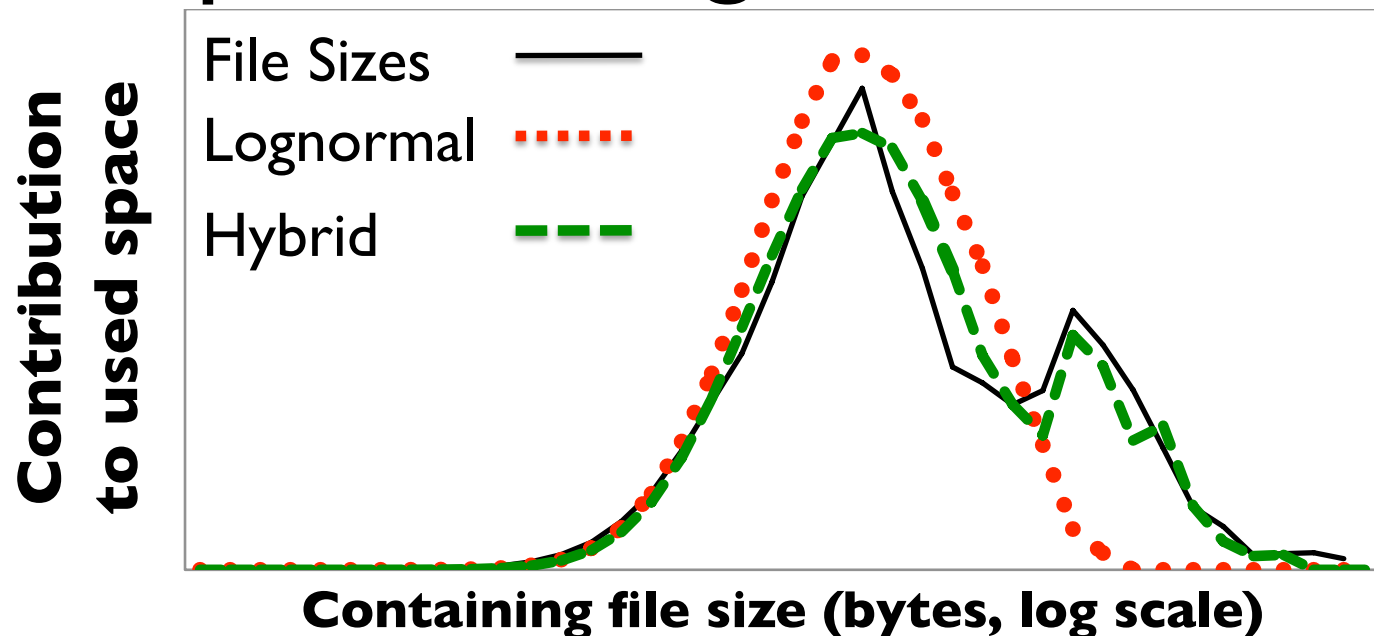
Monte Carlo run

Incorporates dirs by depth  
 and dirs by subdir count

# Creating valid metadata

- Creating file-system namespace
- Creating files: stepwise process
  - File size, file extension, file depth, parent directory
  - Uses statistical models & analytical approximations

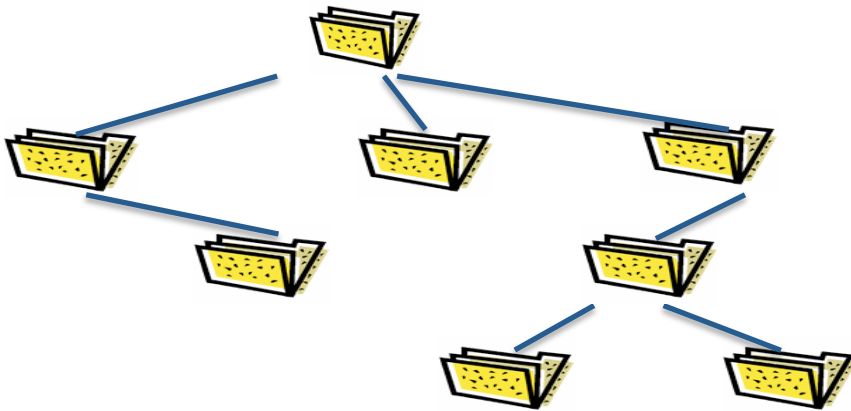
# Example: creating realistic file sizes



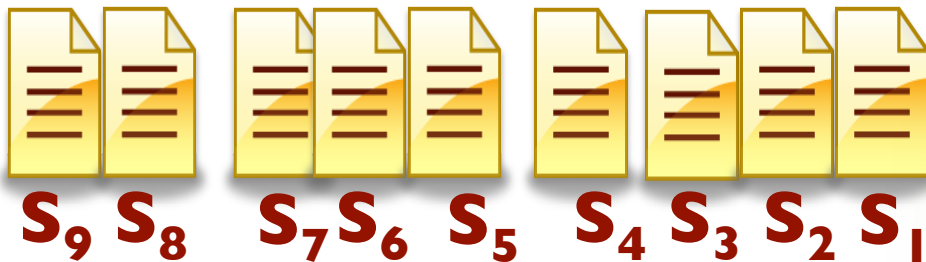
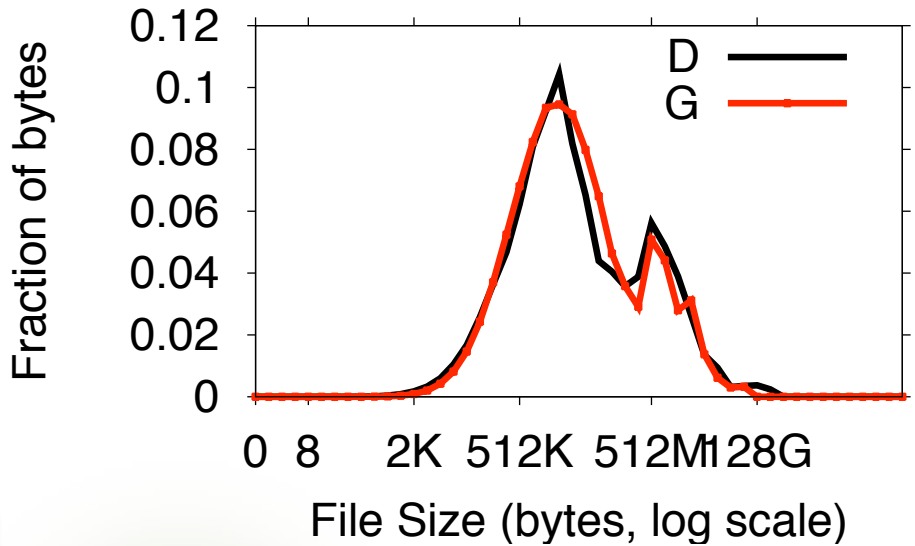
- Pure lognormal distribution no longer good fit
- Hybrid model: lognormal body, Pareto tail
  - Fits observed data more accurately, used to recreate file sizes in Impressions



# Creating files



## Files by containing bytes

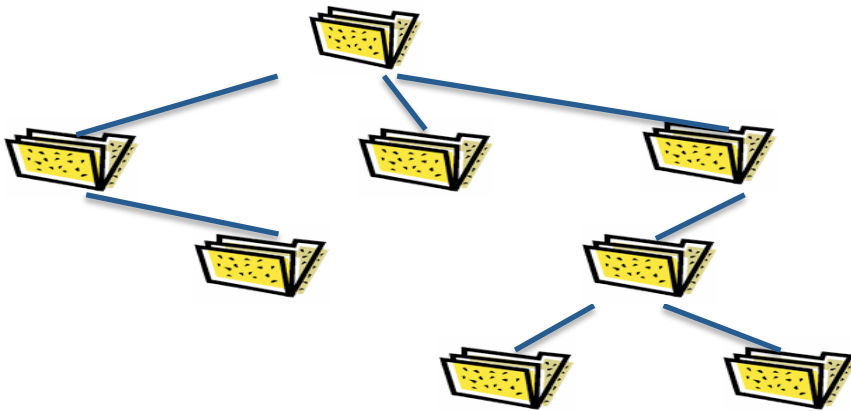


## File Size Model

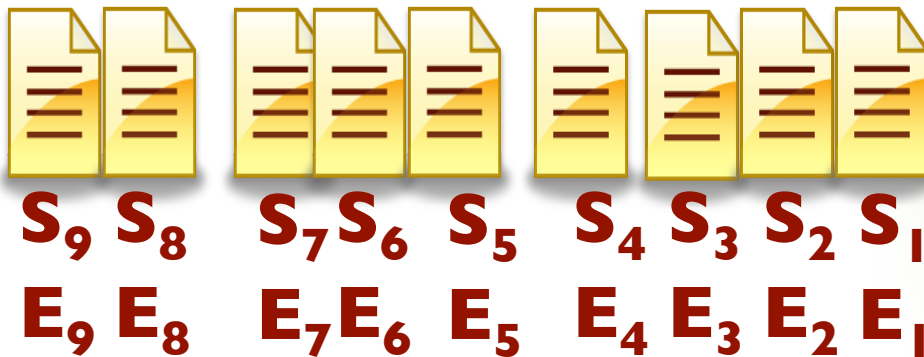
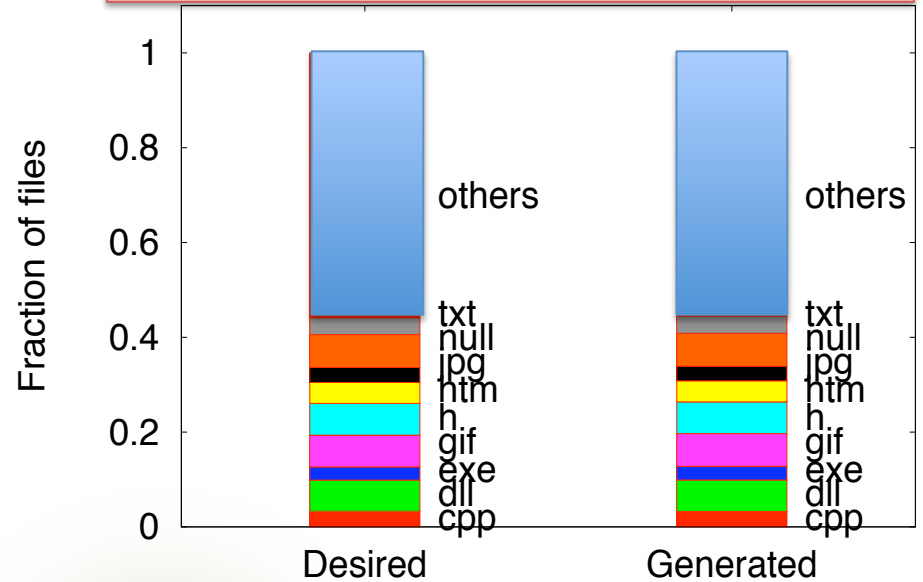
Lognormal body,  
Pareto tail

Captures bimodal curve

# Creating files



## Top extensions by count

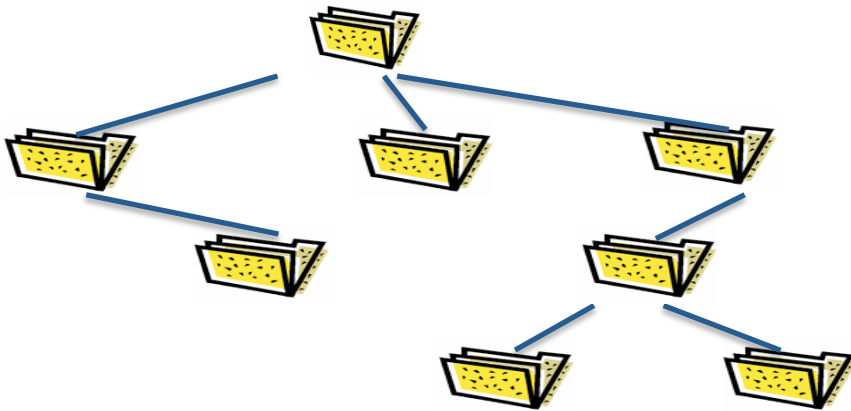


## File Extensions

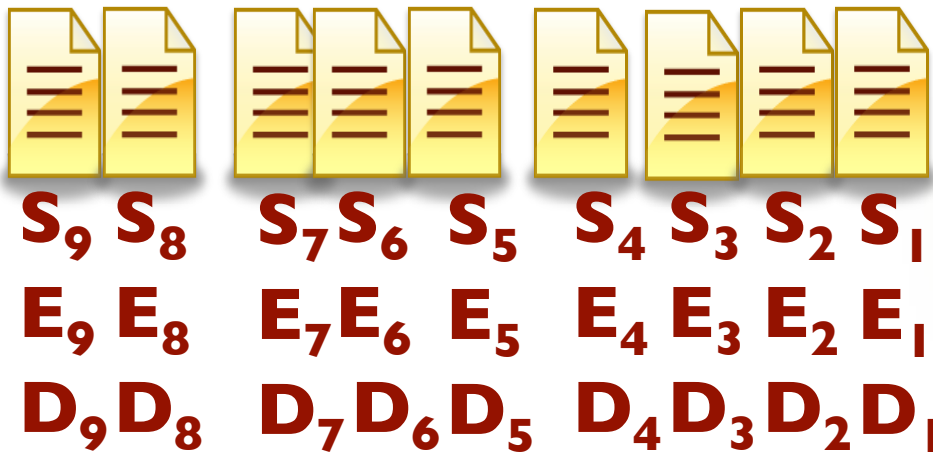
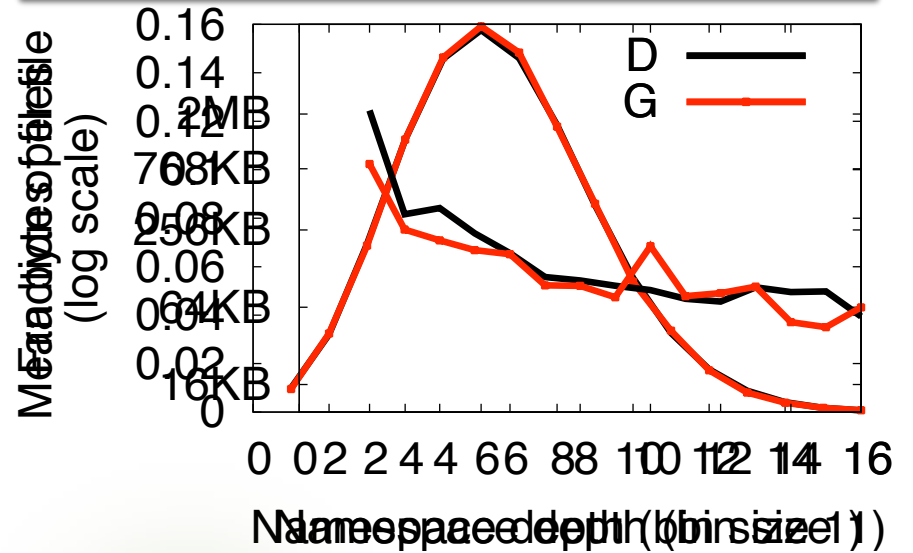
Percentile values

Top 20 extensions account for 50% of files and bytes

# Creating files



Bytes by namespace depth

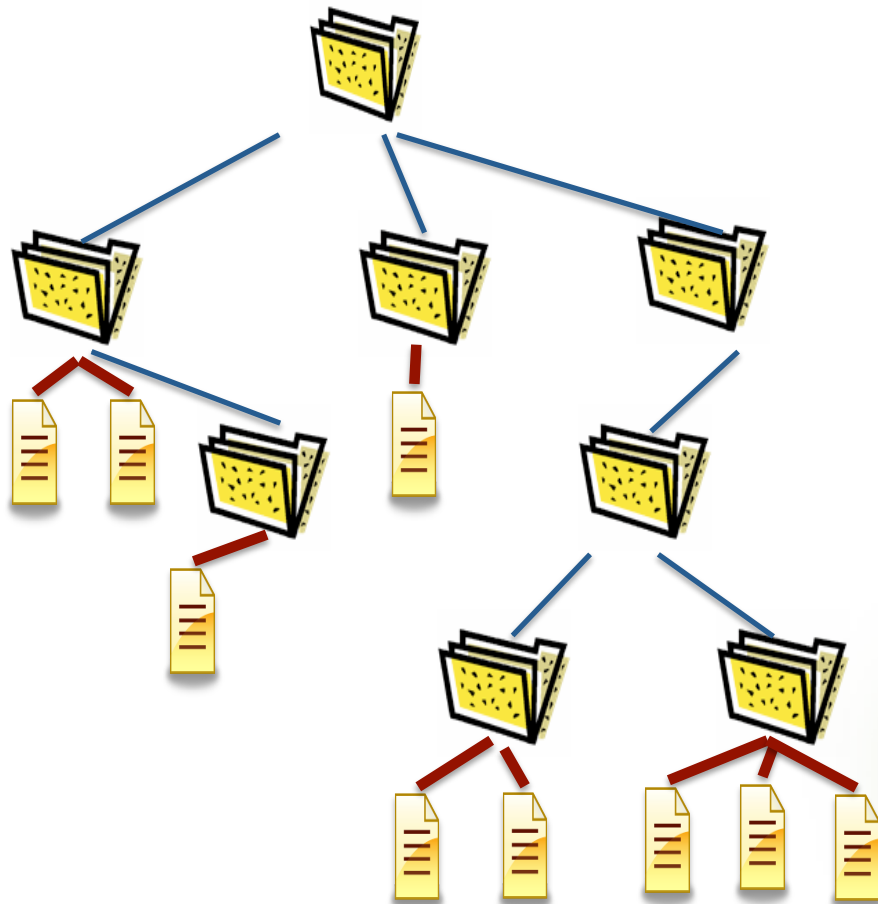


## File Depth

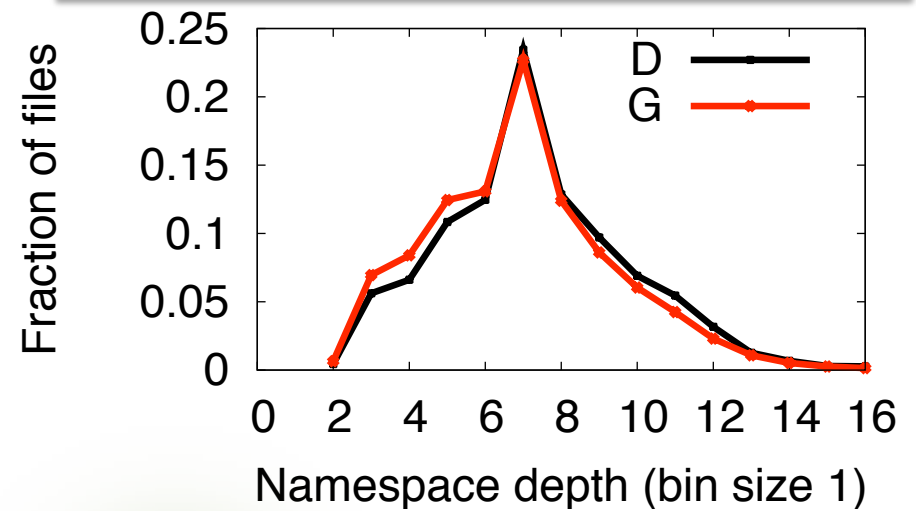
Poisson

Multiplicative model along  
with bytes by depth

# Creating files



Files by namespace depth  
w/ special dirs



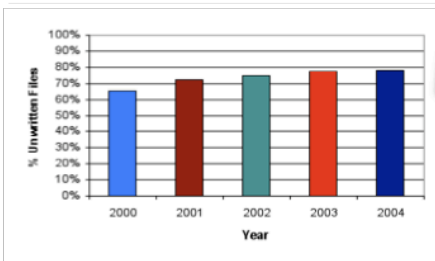
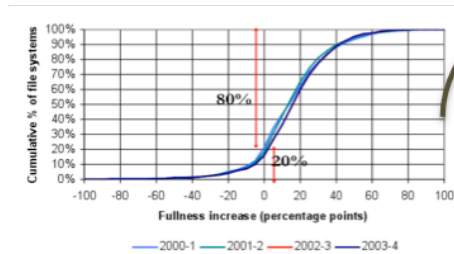
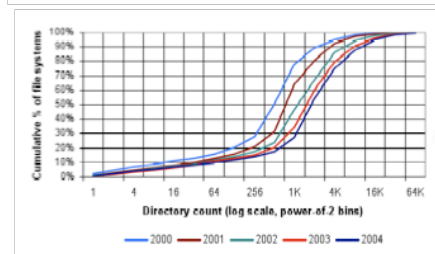
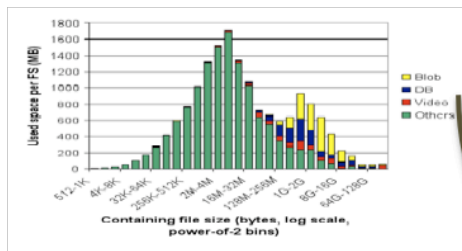
## Parent Dir

Inverse Polynomial

Satisfies distribution of dirs  
with file count

# Resolving arbitrary constraints

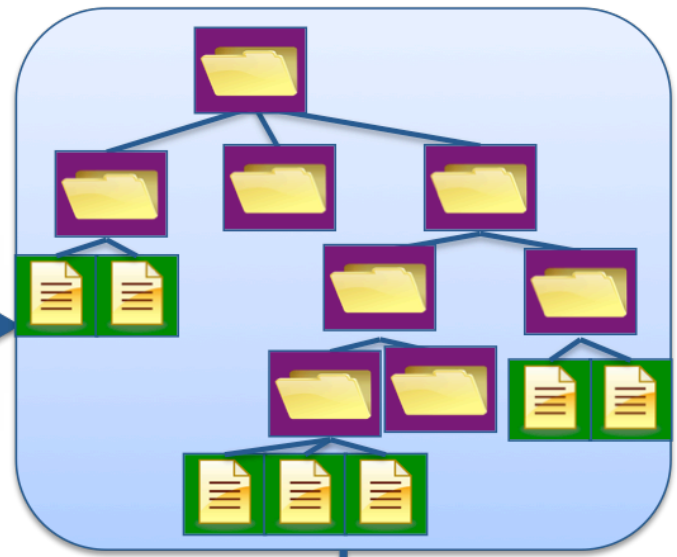
## FS Distributions



User specified FS parameters



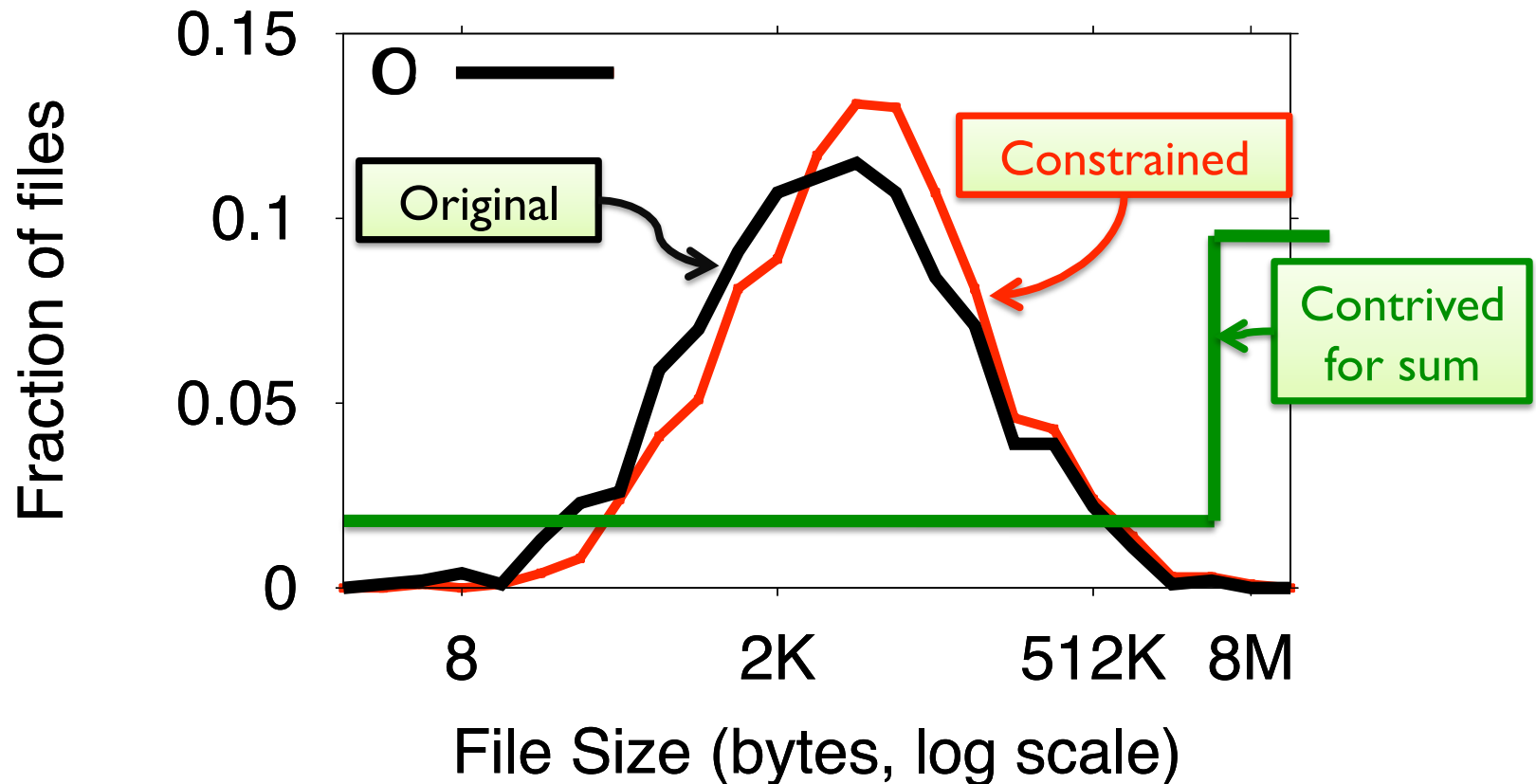
## File-system state



## On-disk state



# Resolving arbitrary constraints



**Constraint:** Given count of files & size distribution, ensure sum of file sizes matches a desired total file system size

# Resolving arbitrary constraints

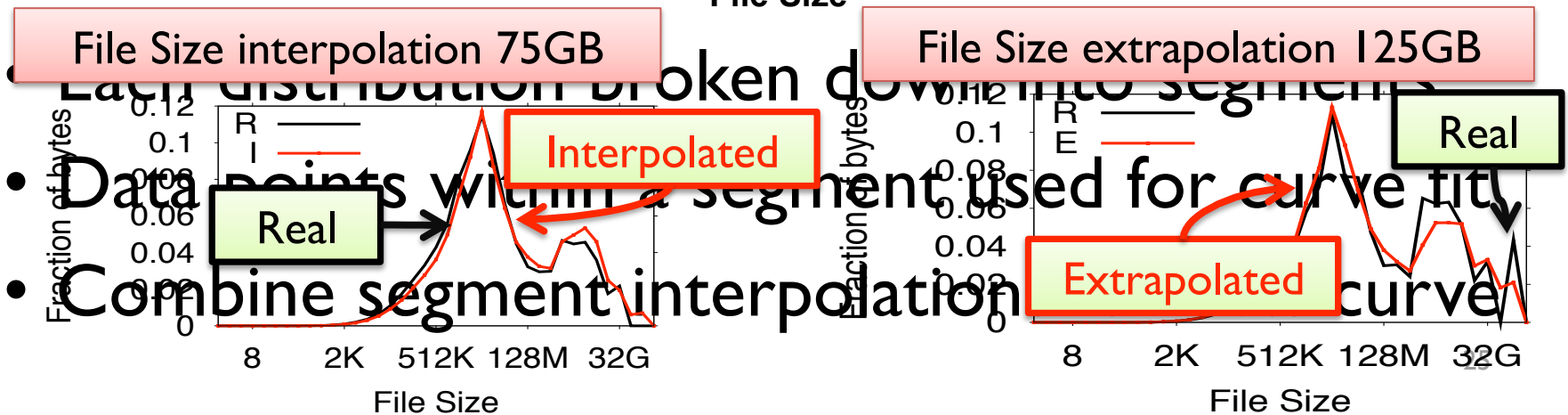
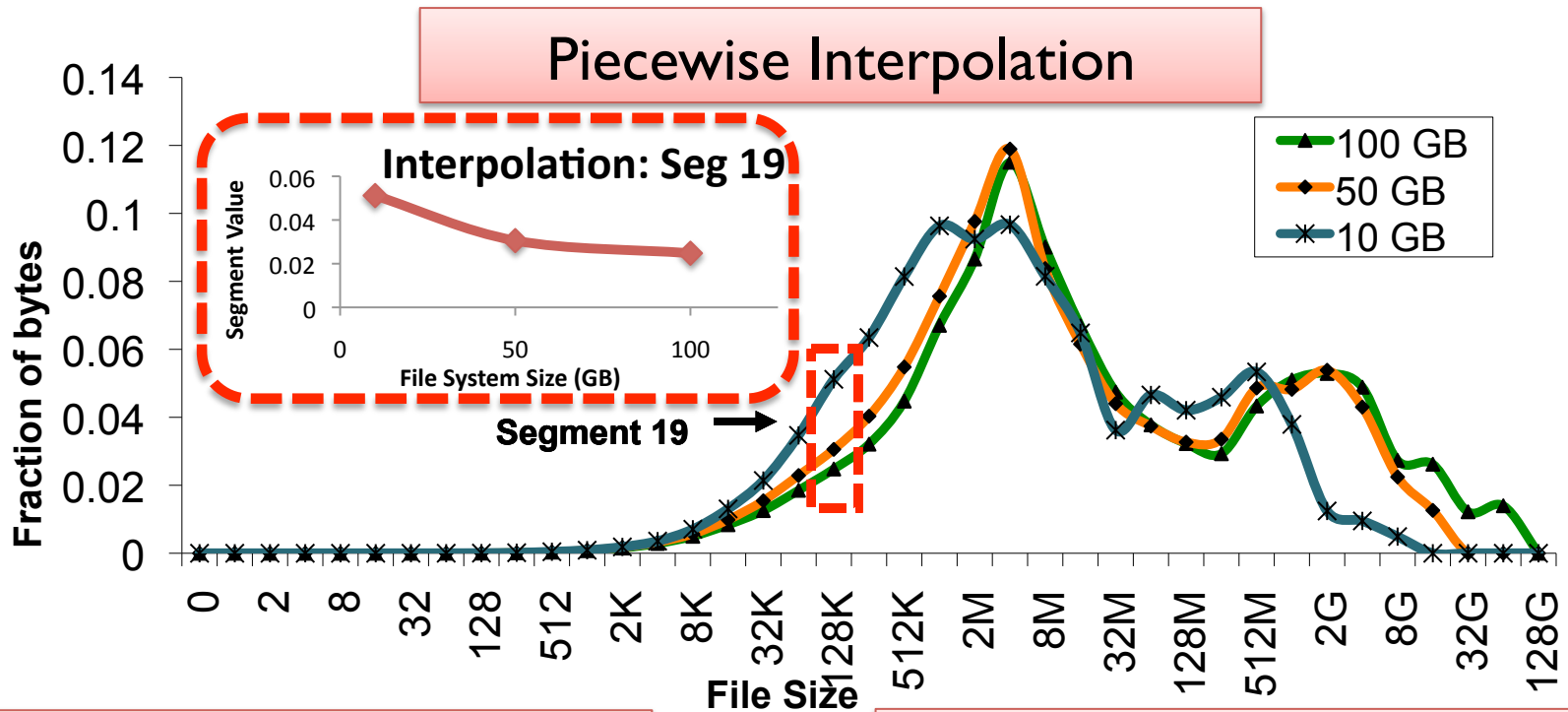
- Arbitrarily specified on file system parameters
- Variant of NP-complete “Subset Sum Problem”
  - Approximation algorithm based solution (in paper)
  - **Oversampling** to get additional sample values
  - Local improvement to **iteratively converge** to the desired sum by identifying best-fit in current sample
- While constraints are satisfied, constrained distribution also retains original characteristics

# Interpolation and extrapolation

- Why don't we just use available data values?
  - Limited to empirical data in input dataset
  - “What-if” analysis beyond available dataset
  - Efficient to maintain compact curve fits and use interpolation/extrapolation instead of all data
- **Technique: Piecewise interpolation**



# Interpolation technique & accuracy

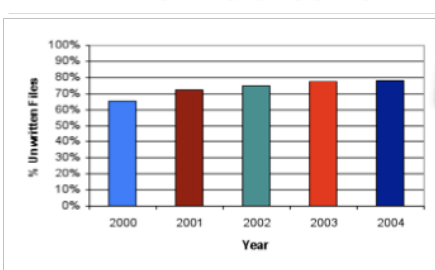
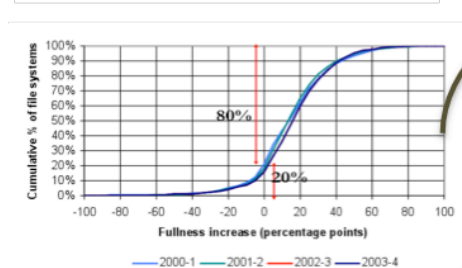
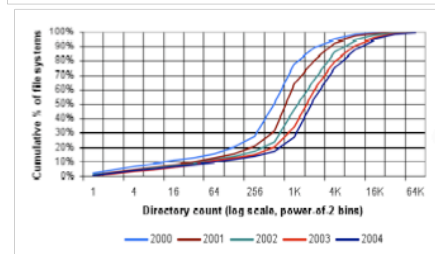
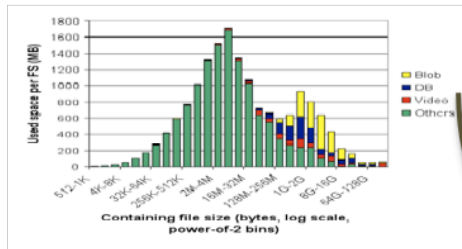


# File content

- Files having natural language content
  - Word-popularity model (heavy tailed)
  - Word-length frequency model (for the long tail)
- Content for other files (mp3, gif, mpeg etc)
  - Impressions generates valid header/footer
  - Uses third-party libraries and software

# Disk layout and fragmentation

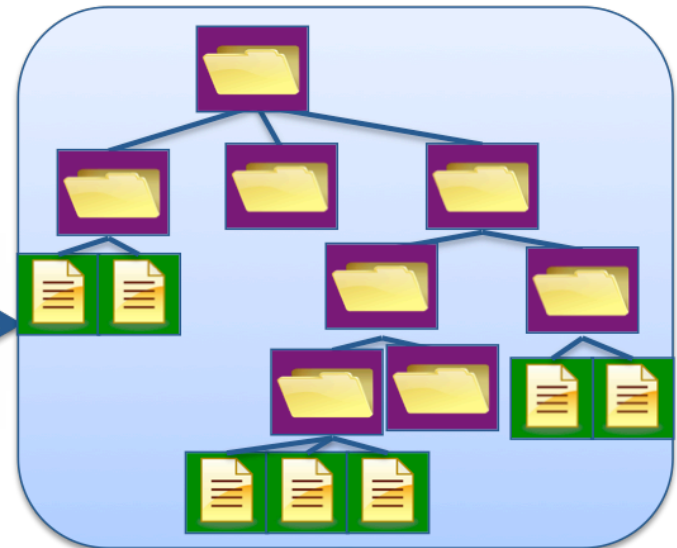
## FS Distributions



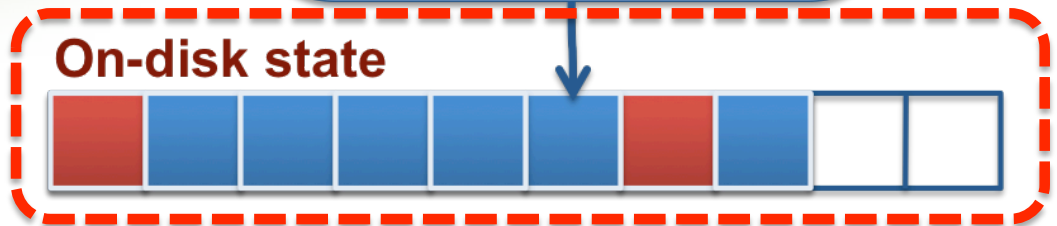
User specified FS parameters



File-system state



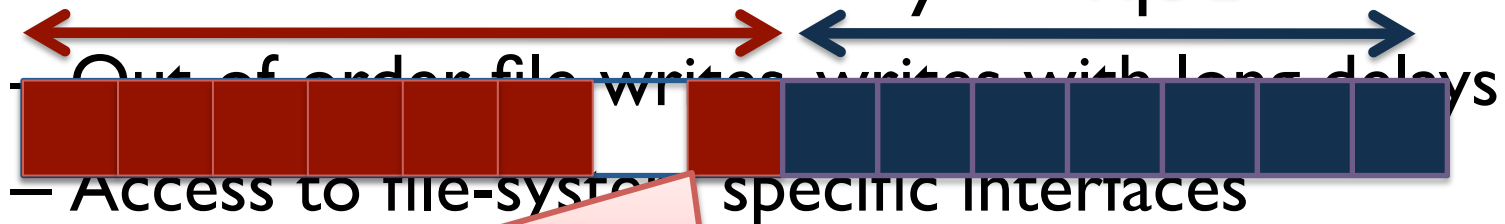
On-disk state



# Disk layout and fragmentation

- Simplistic technique
  - **Layout Score** for degree of fragmentation [Smith97]
  - Pairs of file create/delete operations till desired layout score is achieved

- In future more nuanced ways are possible



1 non-contiguous block (out of 8)

**File Layout Score = 7/8**

All blocks contiguous

**File Layout Score = 1 (6/6)**

# Outline

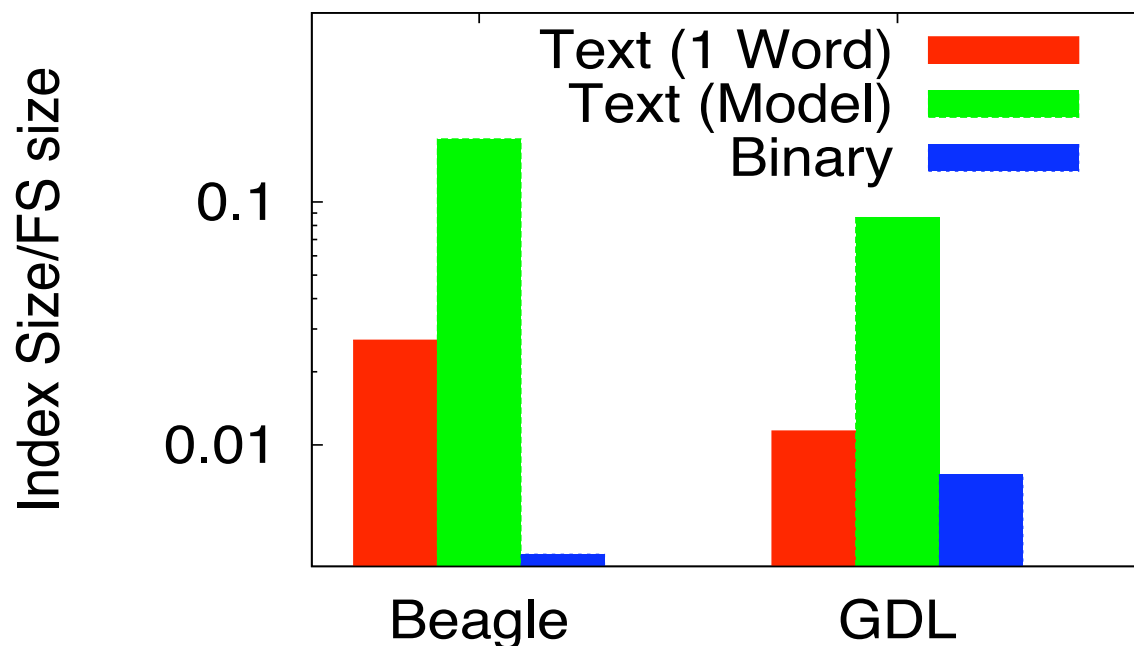
- Introduction
- Generating realistic file-system images
- **Applying Impressions: Desktop search**
- Conclusion

# Applying Impressions

- Case study: desktop search
  - Google desktop for linux (GDL) and Beagle
  - Metrics of interest:
    - Size of index, time to build initial search index
  - Identifying application bugs and policies
    - GDL doesn't index content beyond 10-deep
- Computing realistic rules of thumb
  - Overhead of metadata replication?

# Impact of file content

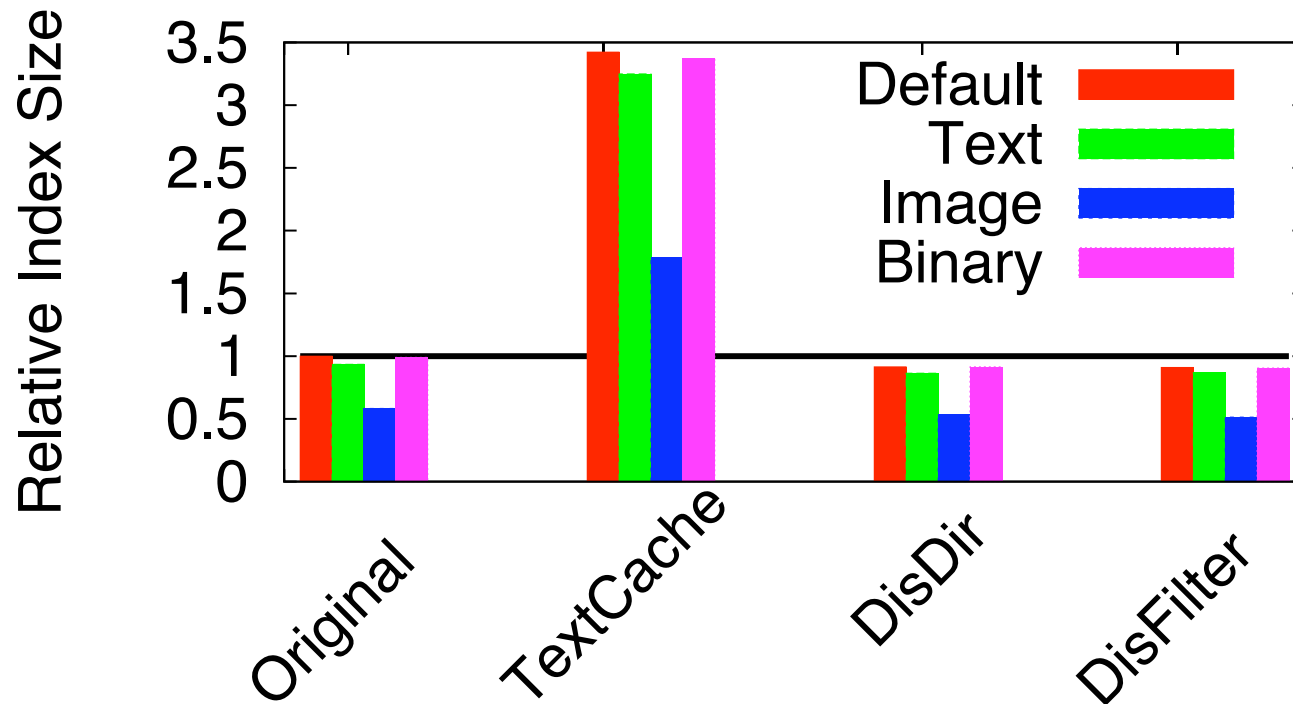
Index Size Comparison



**Understanding design:** GDL index **smaller** than Beagle for text files, **larger** for binary files

# Impact of metadata and content

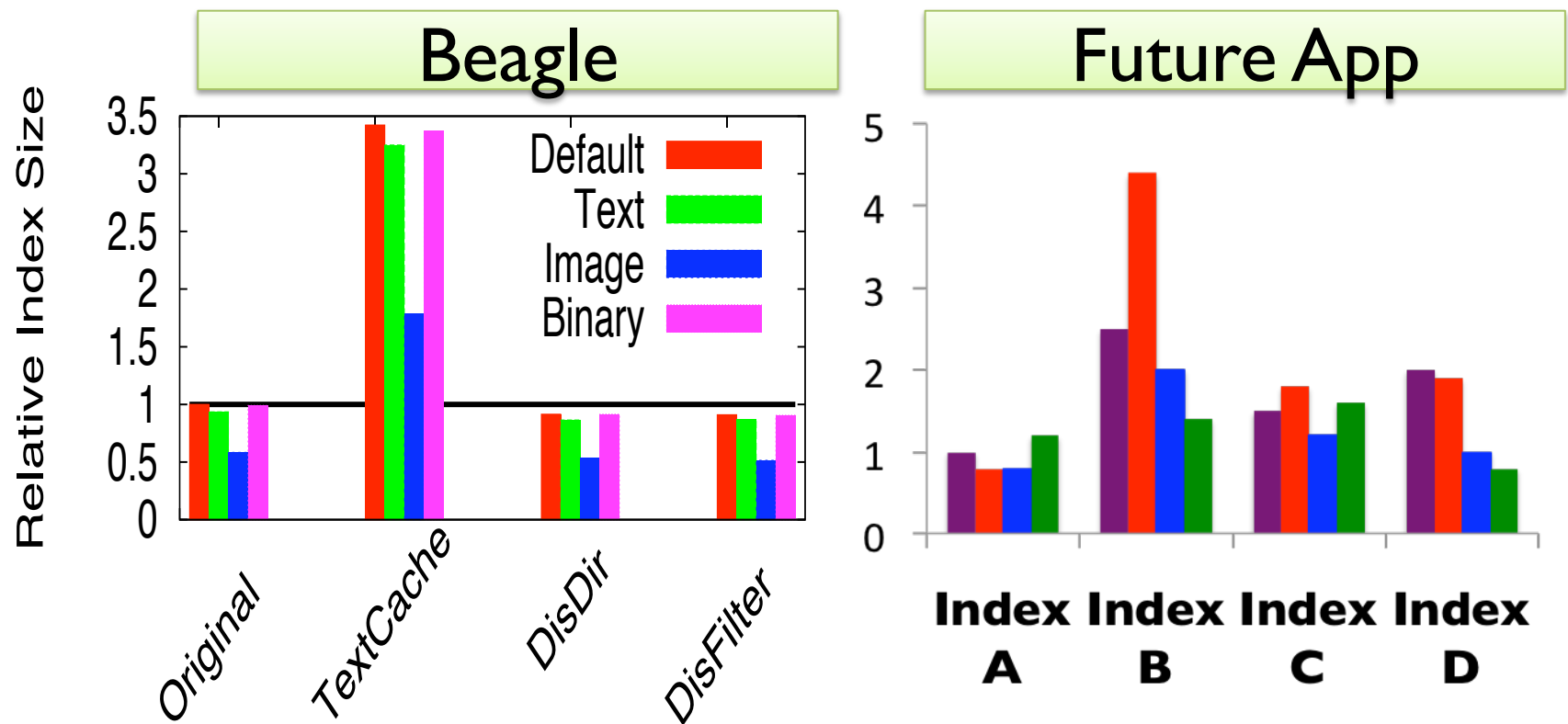
Beagle: Index Size



**Developer aid:** understanding impact of different file system content & different index schemes



# Impact of metadata and content



Reproducing identical file-system image to compare other apps or ones developed later

# Conclusion

- Impressions framework for realistic FS images
    - Representative, controllable, reproducible, easy to use
    - Includes almost all file system params of interest
  - Extensible architecture
    - Plug in new statistical constructs, new models for metadata and content generation
  - Powerful utility for file-system benchmarking
    - To be contributed publicly (coming soon)
- <http://www.cs.wisc.edu/adsl/Software/Impressions>

# Questions?

Nitin Agrawal

<http://www.cs.wisc.edu/~nitina>



Impressions download (coming soon)

<http://www.cs.wisc.edu/adsl/Software/Impressions>

