Semantic Data Placement for Power Management in Archival Storage

Avani Wildani & Ethan L. Miller
Storage Systems Research Center
Center for Research in Intelligent Storage
University of California, Santa Cruz

NSF Center for Research in Intelligent Storage

What is archival data?

- Tape back-ups
- Compliance records
  - Sarbanes-Oxley
  - Government correspondence
- Abandoned experimental data
- Outdated media
- “Filed” documents
- Vital records

Mission

- Save power in archival systems
  - Disks incur the highest power cost in a datacenter
  - As disks get faster, power grows as a square
- We can save power by reducing the number of spin-ups in archival systems
  - Spin-ups can consume ~25x the power of idling
  - Spin-ups reduce device lifetime

Saving power

- Power management in archival storage typically relies on having few reads
  - Modern, crawled archives can't make this assumption
- Steady workload types can be exploited
  - 30% hit rate gives ≥ 10% power savings
    - Hits: reads that happen on spinning disks

“Archival by accident”

- Hundreds of exabytes of data are created annually
  - Flickr, blogs, YouTube, ...
- “Write once / Read-maybe” may not hold
  - Search indexers
  - Working set changes
- Web has archival characteristics
  - Top 10 websites account for 40% of accesses*
  - Drop off is exponential, not long tail
- Much data becomes **archival by accident**

*The Long Tail Internet Myth: Top 10 domains aren’t shrinking (2006)
http://blog.compete.com/2006/12/19/long-%20tail-chris-anderson-top-10-domains

Big Idea

- Fragmentation on a disk causes a significant drop in performance
- “Fragmentation” of a group of files that tend to be accessed together across a large storage system is similarly bad
- Defragmentation is hard, but we should at least try to append onto groups where we can!

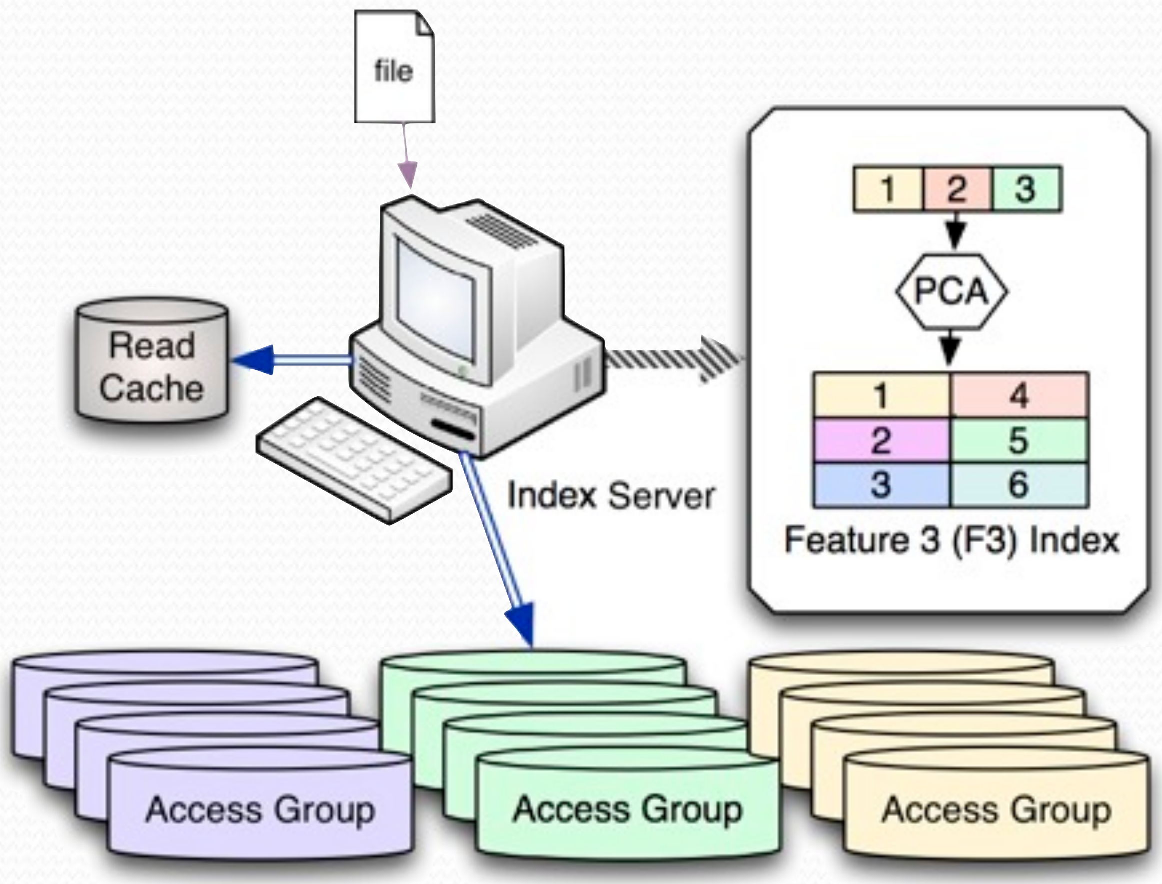
Overview of our method

1. Storage system is divided into *access groups*
2. Files likely to be accessed together are placed together into an access group
3. When a file in an access group is accessed:
   3.1. Its disks are spun up
   3.2. The disks are left on for a period of time *t* to catch subsequent accesses

- Goal : Save power by avoiding repeated spin-ups

System design

- Index Server:
  - Classification
  - Cache
- Disks:
  - MAID semantics: usually off
  - Logically arranged into access groups
    - Parity is done over an access group

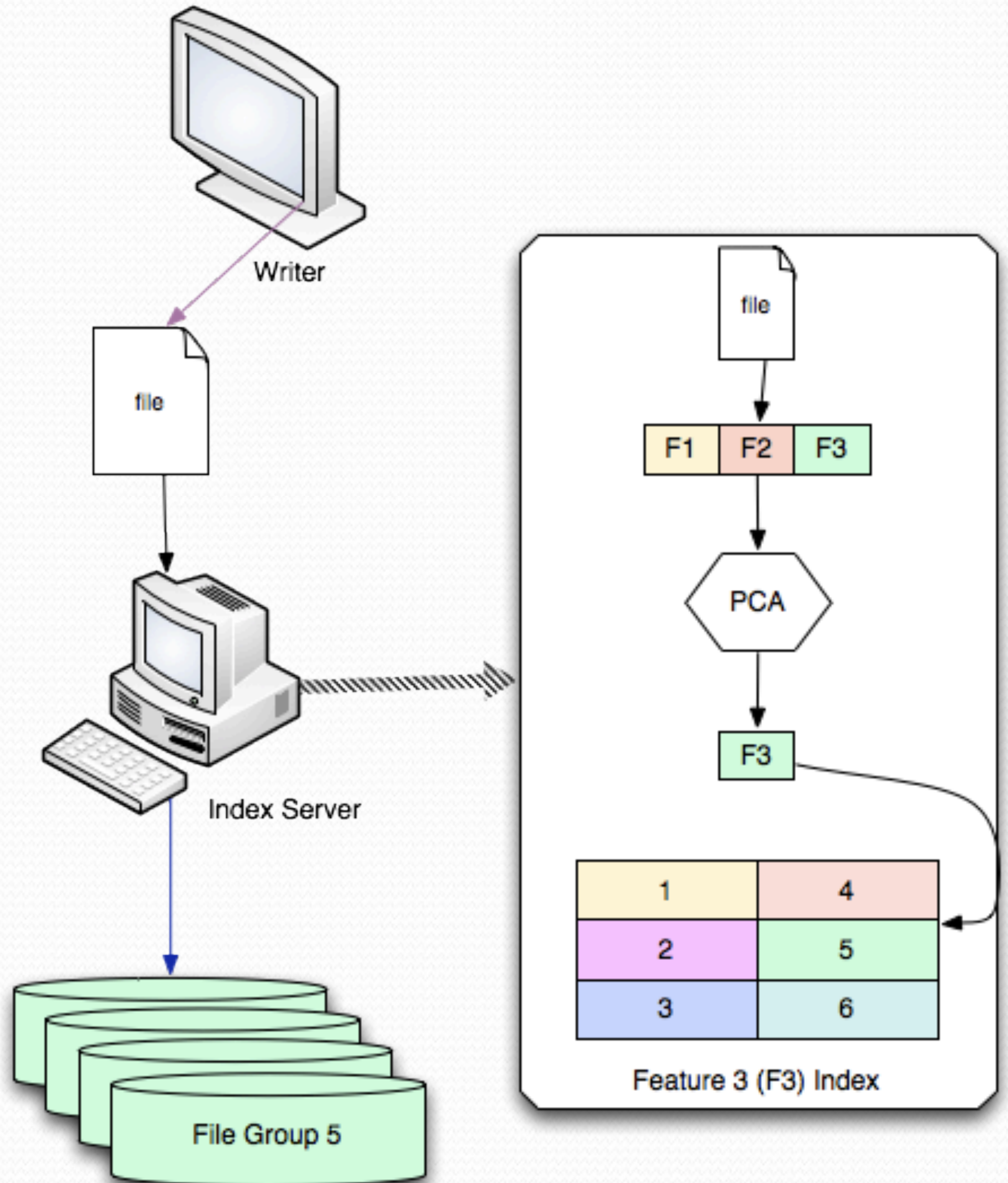


System Design: Bootstrap

- Start with set of data
- Index servers split data into groups
  - Assumption: Classifications will last for system lifetime
  - $O(n^3)$
    - Cheaper, linear methods exist, but...
    - This only has to be done once!
- Stripe data onto access groups
  - Parity is determined by total desired system cost.

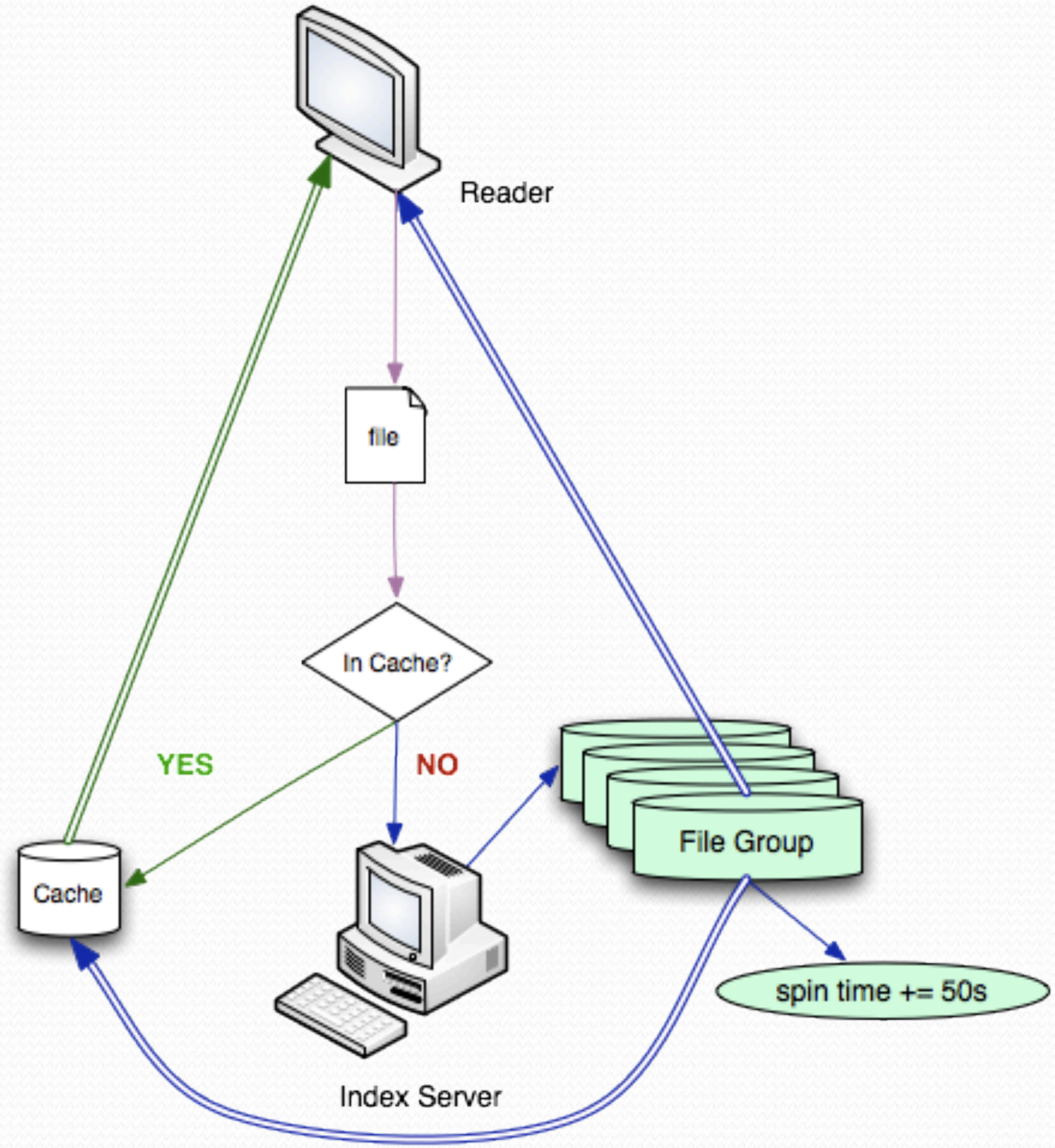
System design: writes

- Writes are batched by default
  - File will write at next spin-up
    - Sooner if write cache fills
- If file group is full, split



System design: reads

- Cache could be simple LRU
- If file group is spinning, add to the spin time
  - Catches subsequent accesses
  - Power is wasted if no subsequent accesses



Splitting an access group

- Access groups will grow as files are added
  - Large access groups lower power gain: split them!
- Large access groups are marked for splitting
  - Wait for next spin-up.
- Groups too small to sub-classify
  - Split randomly
- Could potentially use existing split (e.g., path hierarchy)

Selecting classification features

- Select features to classify with: type, creator, path
  - Frequently meta-data
  - Use labels if provided
- Pick features with principal component analysis
  - "What features matter most in differentiating groups of files"
- Use expectation maximization:
  - Expectation: $Q(\theta|\theta^{(t)}) = \mathrm{E}_{Z|x,\theta^{(t)}}\left[\log L(\theta; x, Z)\right]$
    - Calculate log likelihood for eigenvectors in covariance matrix
  - Maximization: $\theta^{(t+1)} = \underset{\theta}{\arg\max}\ Q(\theta|\theta^{(t)})$
    - Maximize over expectations
    - Re-do expectation step

SSRC CRSS Baskin Engineering UC SANTA CRUZ

Classification

- Without history:
  - Blind source separation
  - tf-idf:

$$\mathrm{tf_{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

$$\mathrm{idf_i} = \log \frac{|D|}{|\{d_j : t_i \in d_j\}|}$$

$$\mathrm{tfidf_{i,j}} = \mathrm{tf_{i,j}} \cdot \mathrm{idf_i}$$

- With history:
  - Hierarchical clustering
    - Make lots of small clusters and progressively combine them
  - Access prediction
    - Learn what is likely accessed together
    - Create a dynamic Bayesian network

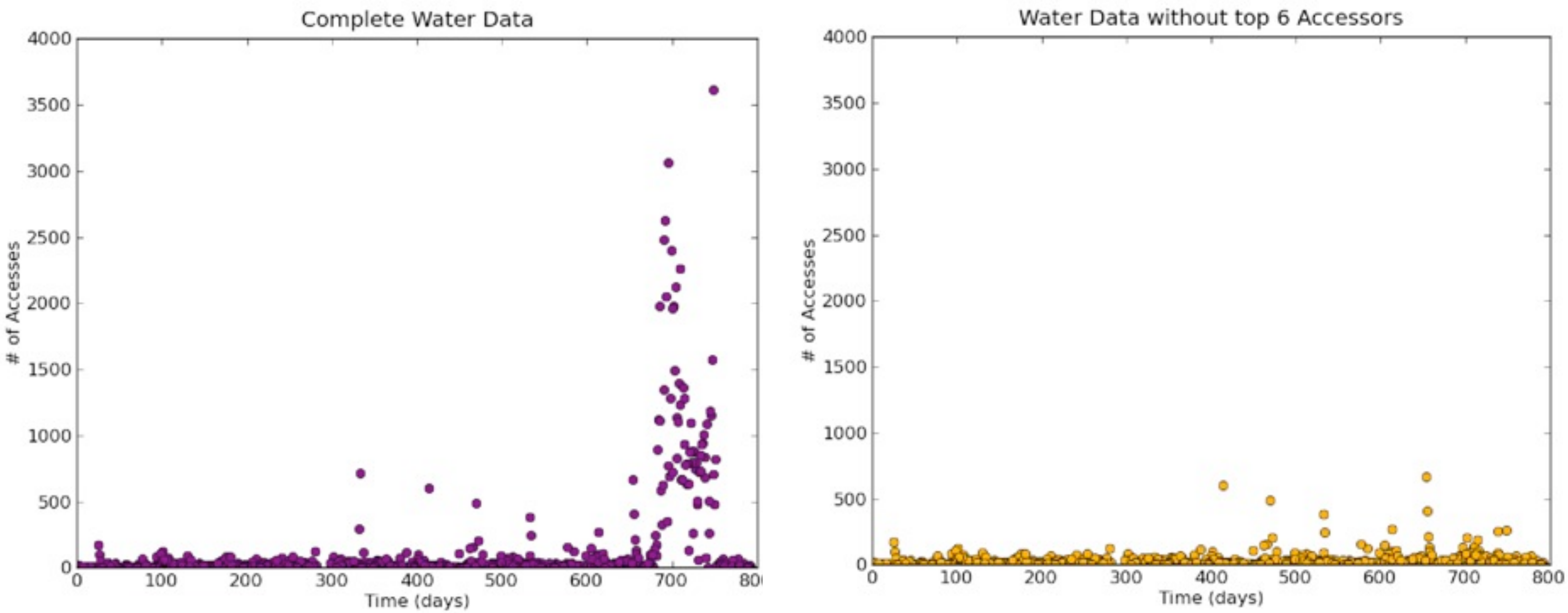
Definitions

- Hit Rate: % of reads that happen on spinning disks
- Singletons: % of reads that result in a spin-up with no subsequent hits within $t = 50$ seconds
- Power Saved: % of power saved vs. paying one spin-up cost for every read

Data sets

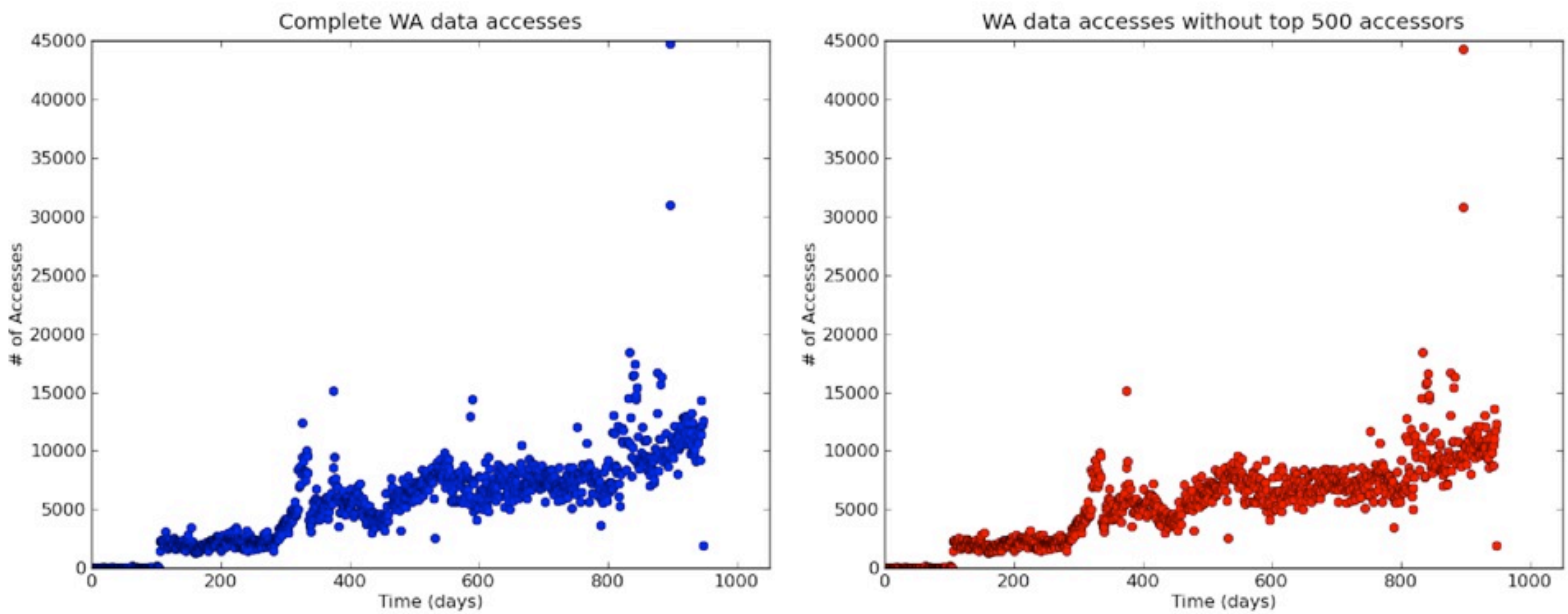
- Web access logs for a water management database (DWR)
  - ~90,000 accesses from [2007-2009]
  - 2.3 GB dataset
  - Accesses come pre-labeled with features
    - E.g. Site, Site Type, District
- Washington State records (WA)
  - ~5,000,000 accesses from [2007-2010]
    - Accesses are for retrieved records
  - 16.5 TB dataset
  - Single category, pre-labeled

Access frequencies: DWR



- Search indexers can cause significant spikes in archival access logs

Access frequencies: WA

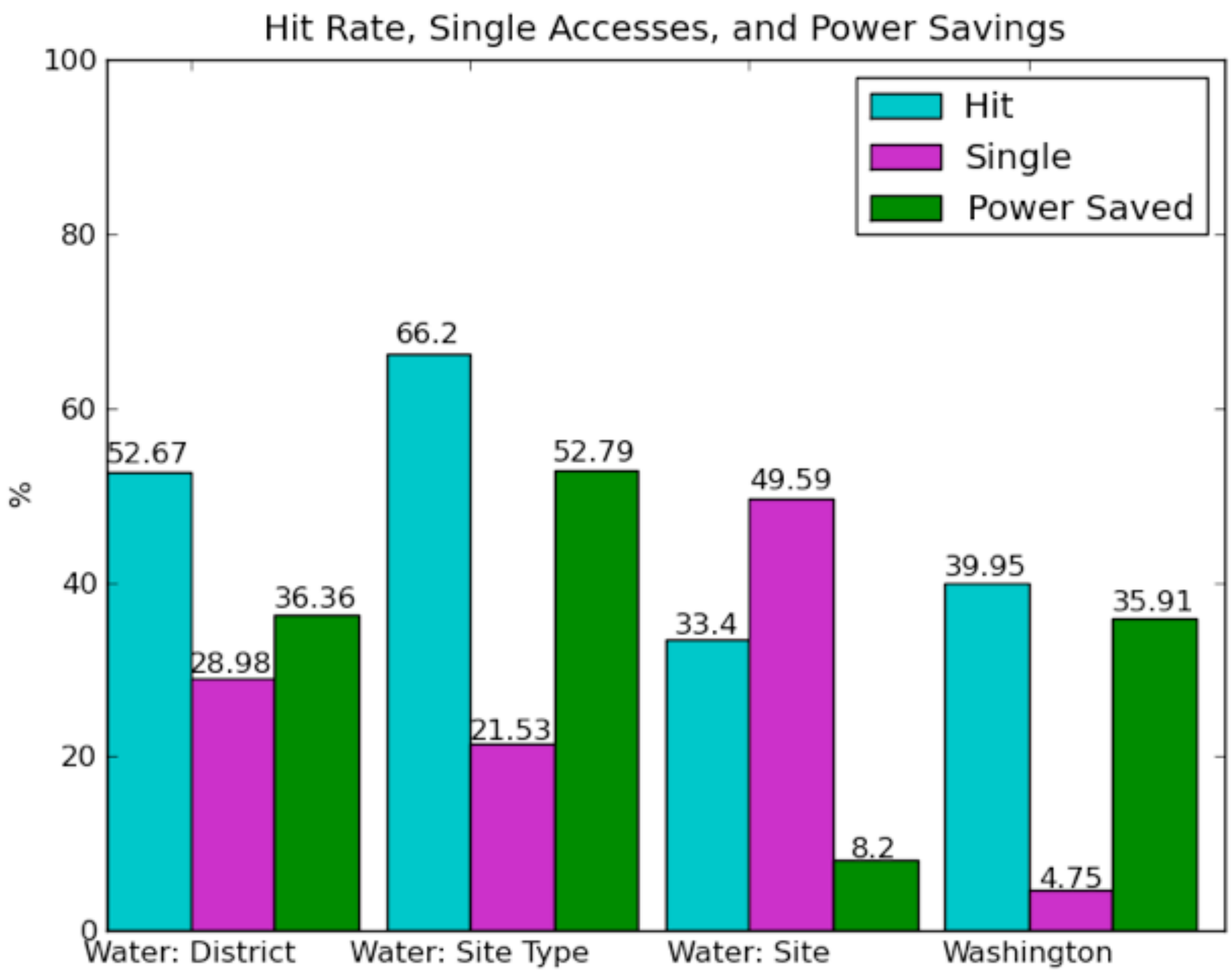


- Spikes can appear without a clear culprit

How can we group the DWR data set?

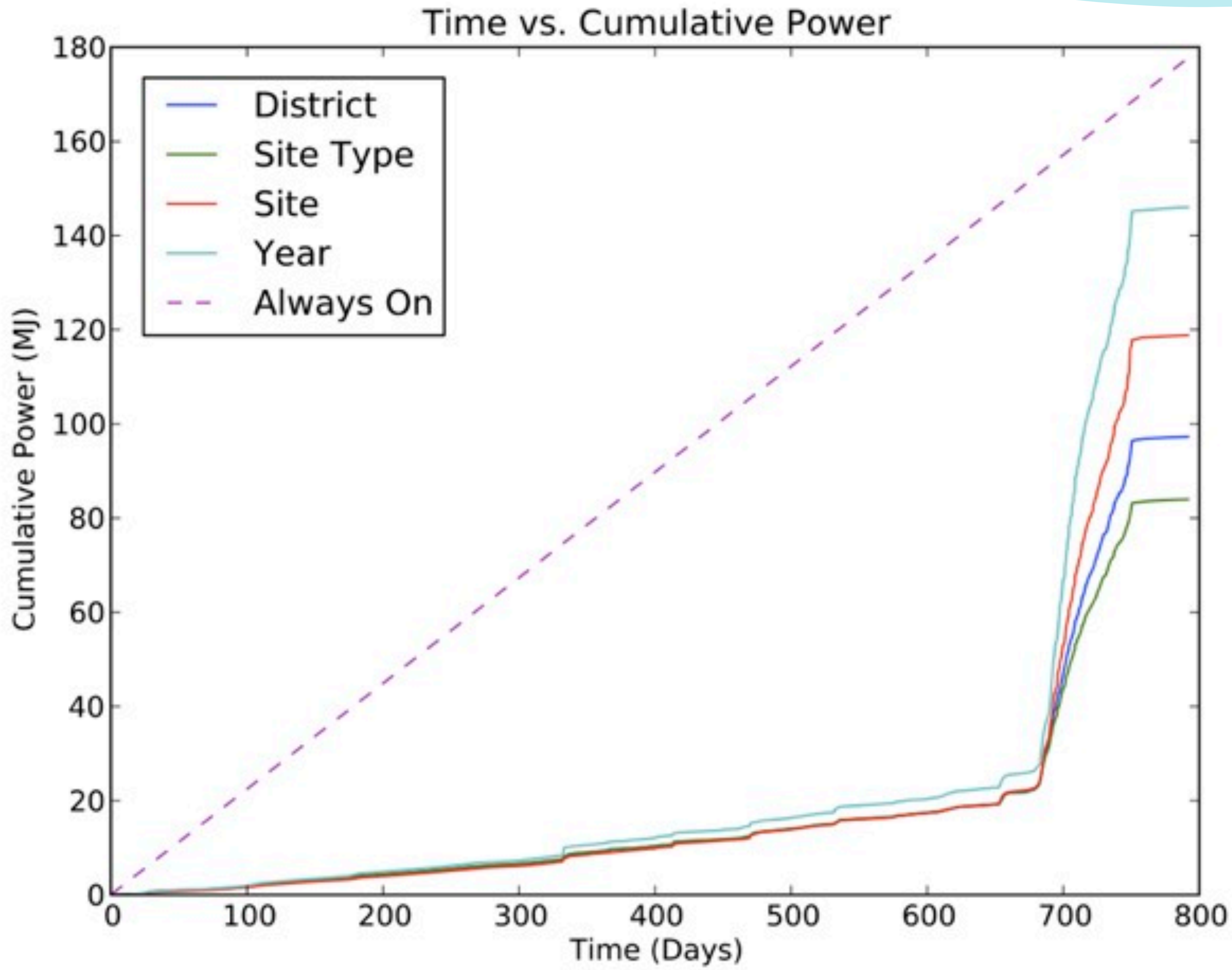
- Clustering is difficult because the directory structure isn't exposed
  - We can automatically infer 'Site'
- Some water files can be parsed to detect signatures
  - Not generally applicable

Power savings



- Power savings is strongly dependent on singletons
- Hit rate is >30% for all datasets

Grouped vs. always on

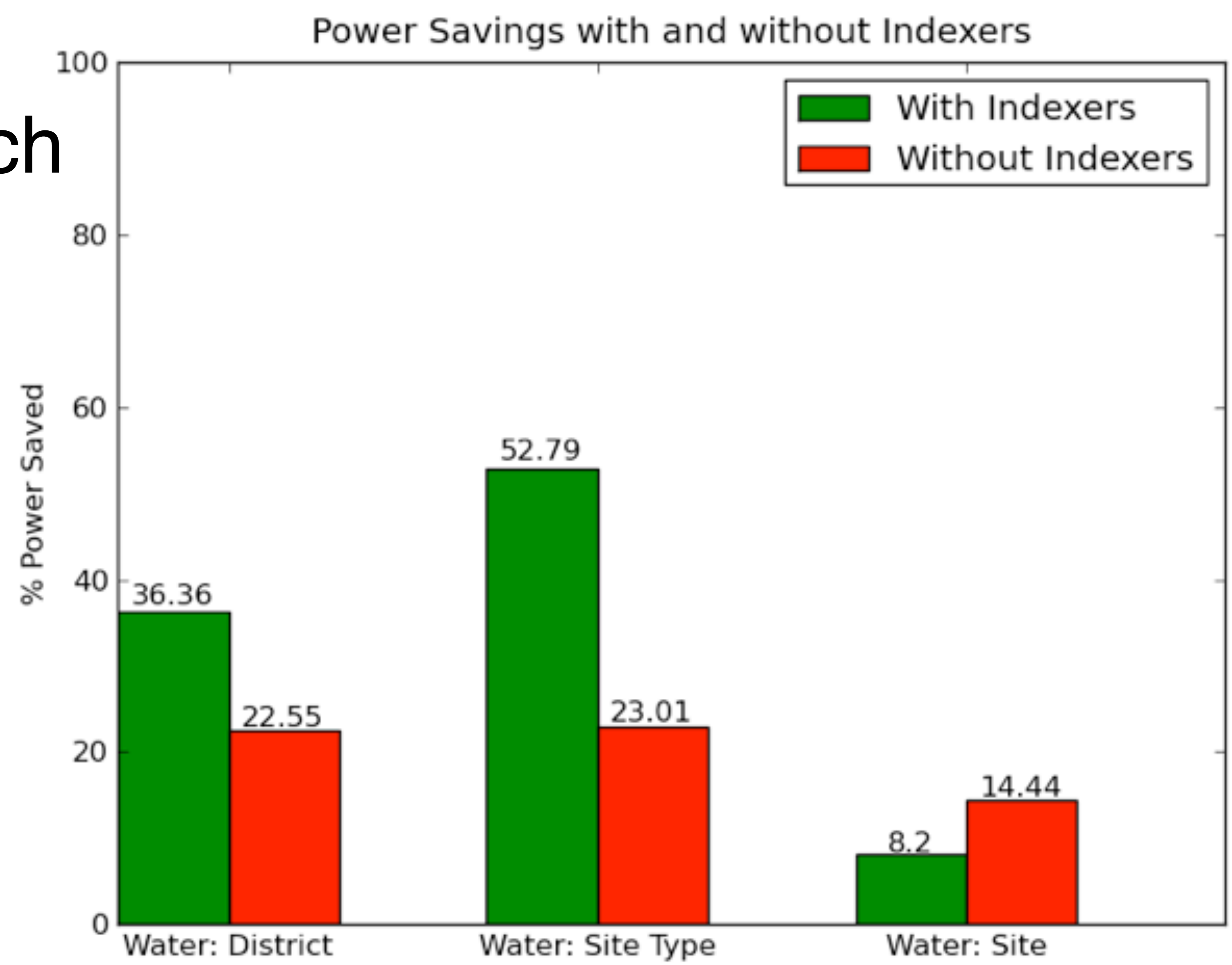


- All our groupings save more power than leaving all disks on
- Spike is from indexers

Effect of search indexers

- Search indexers can alter feature importance

- Site subgroup: Search indexers can create singletons



Future work

- Failure isolation
- Refined grouping
- Caching entire active access group
- Re-allocation of access groups
- SLO / priority implementation
- More data sets

Summary

- Files used all the time don't impact rest of archival system power footprint
- Real data has enough closely consecutive accesses to save power (30–60%)
  - Range indicates we could do better
- Grouping data saves significant power (up to 50%)
- Archival-by-accident systems are a growing research area

Questions?

Please come talk to me if you have
I/O traces from **archival systems**

Thanks to Ian Adams for help with the traces!

Thanks to our sponsors:

STORAGE SYSTEMS RESEARCH CENTER

NSF Center for Research in Intelligent Storage