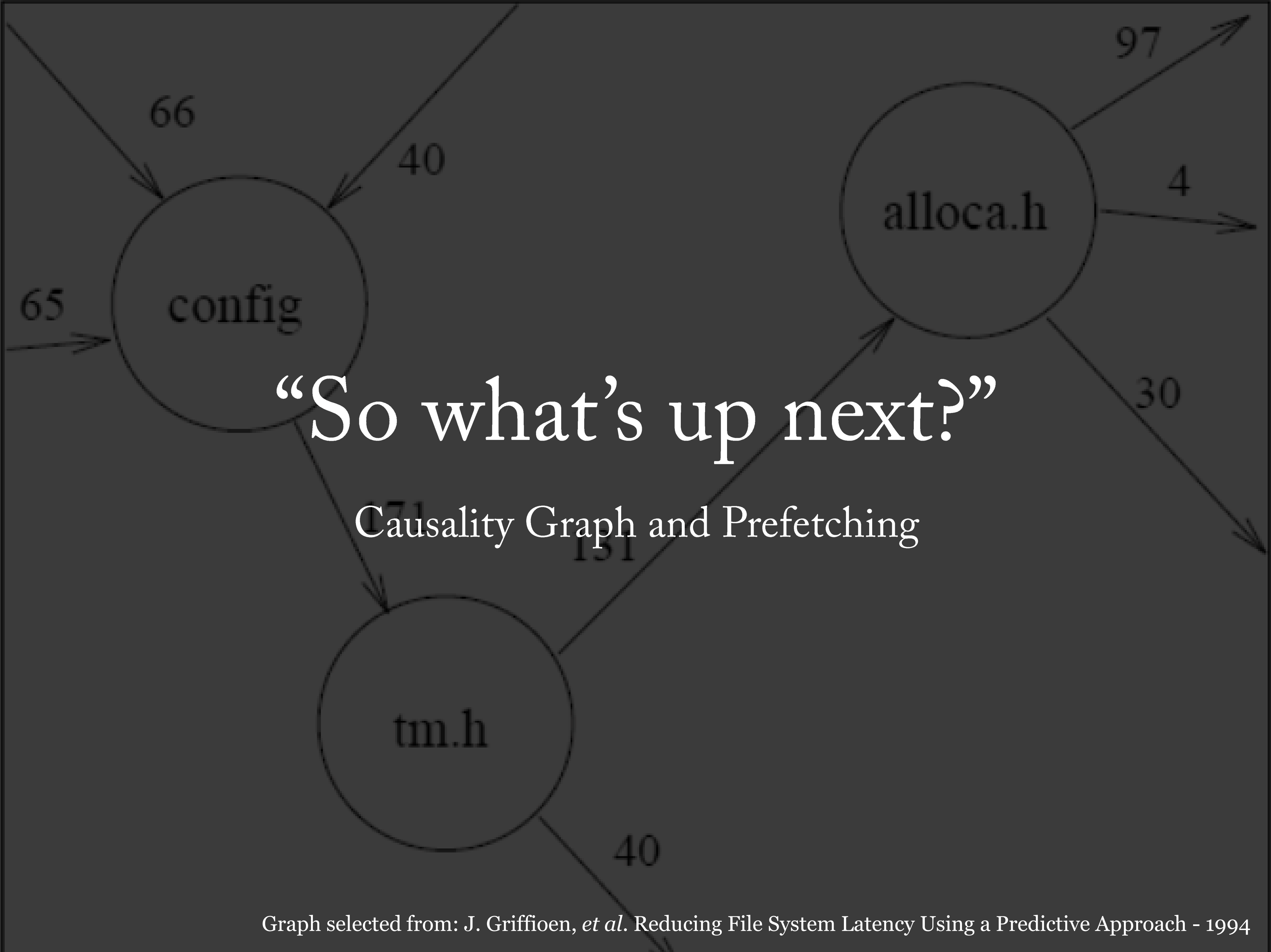


“So what’s up next?”

Causality Graph and Prefetching

Graph selected from: J. Griffioen, *et al*. Reducing File System Latency Using a Predictive Approach - 1994

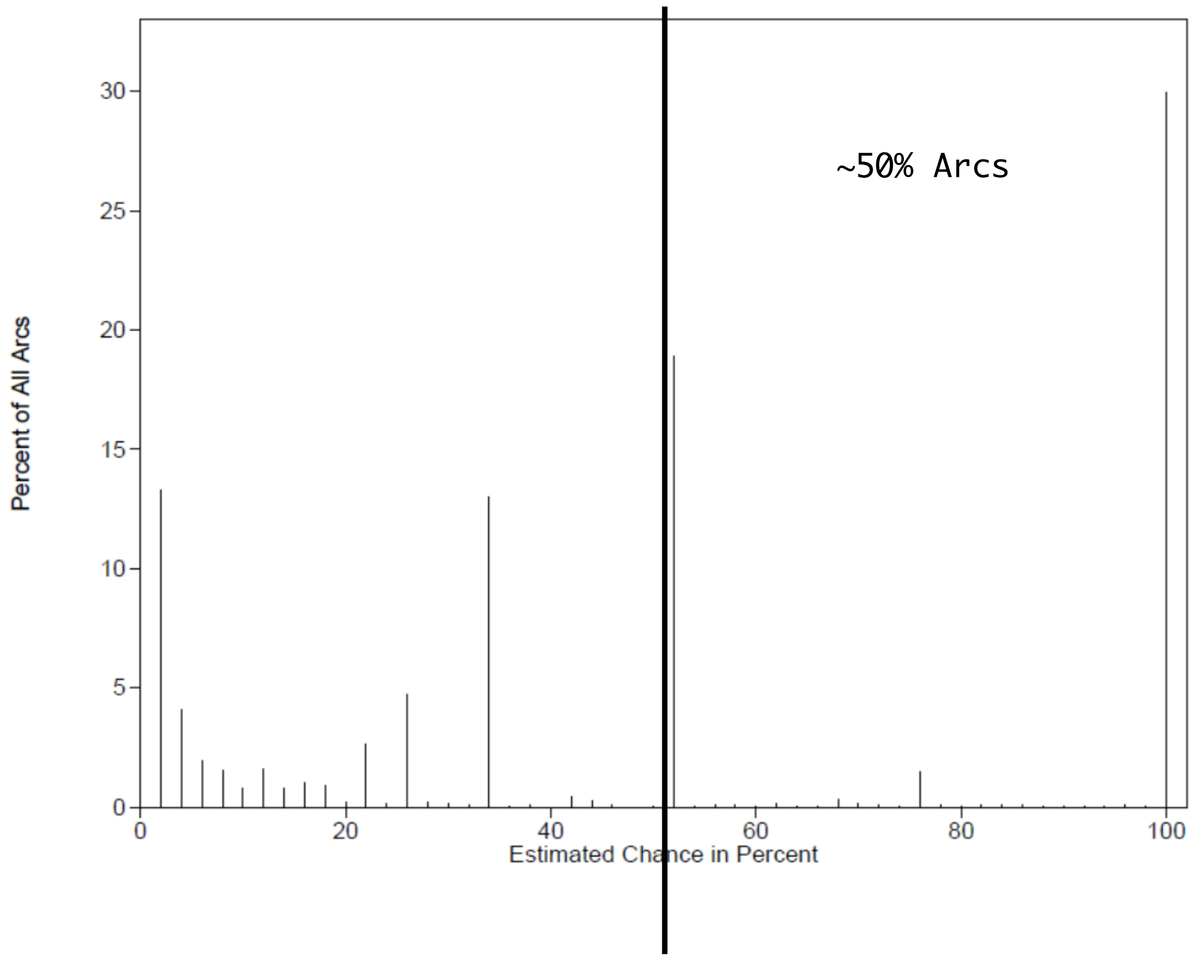
Traditional Caching – Not enough

- LRU cache "knows" nothing
  - Some stuff used for only one time take up space
  - Those not in cache might be requested soon
- Need to know what happened before
- (Automatic) Prefetching
  - Predicts future file requirements from past activity

Prefetching Files - 1994

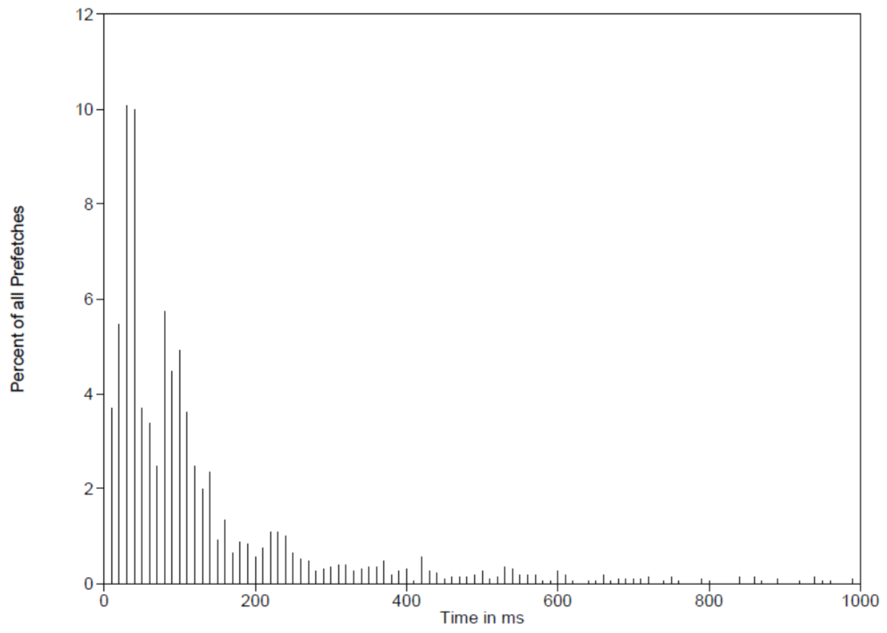
- J. Griffioen & R. Appleton - Reducing File System Latency Using a Predictive Approach
- Created a probability (causality) graph
- Lookahead period – file operating syscalls
  - File B opens “soon after” file A = B is related to A
- Chance of correct prediction = $P[B \mid A]$
- Minimum chance
  - System resources were limited at that time

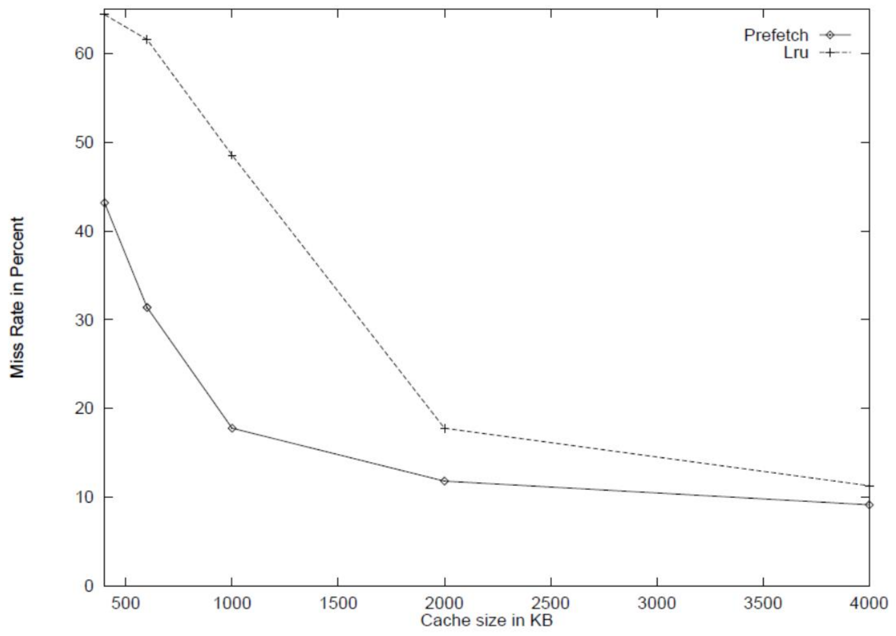
Prefetching Files - 1994



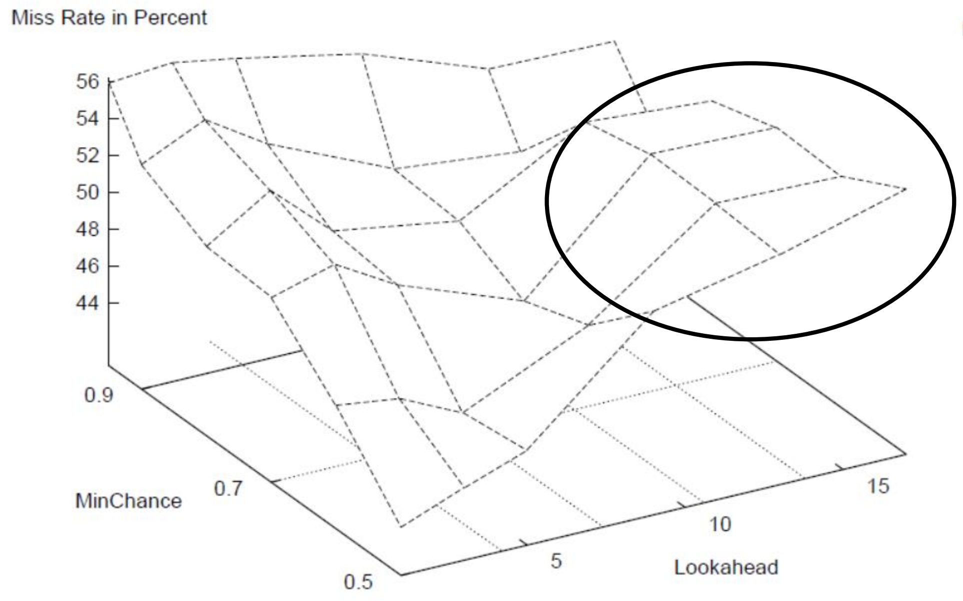
$\geq 50\% \rightarrow$

Prefetching Files - 1994

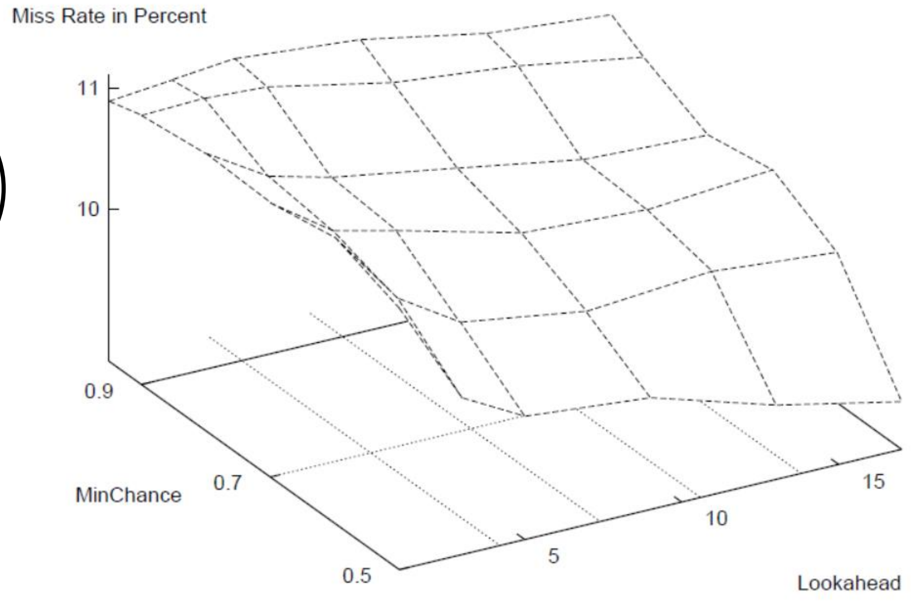




Prefetching Files - 1994



Small - 400 KB



Large - 4000 KB

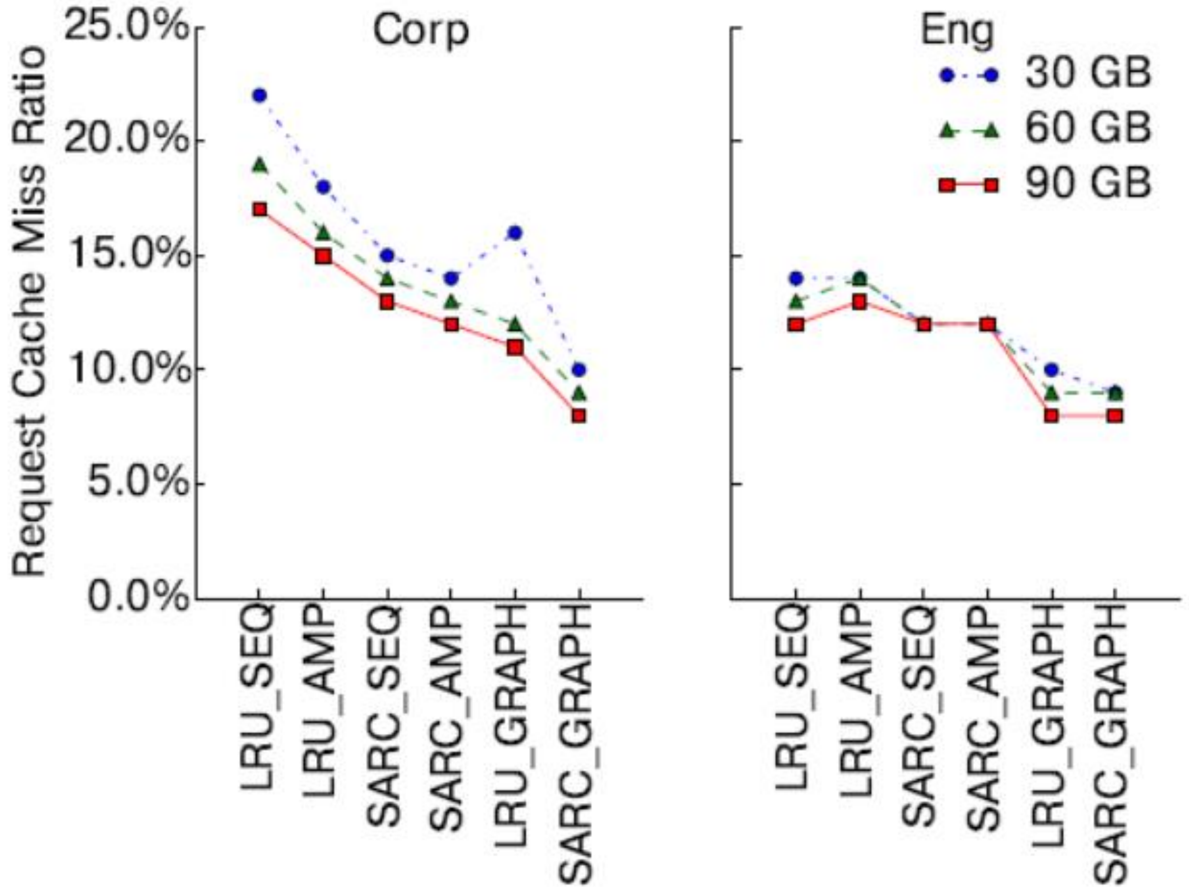
Prefetching Files - 1994

- Look-ahead may imply (but not guarantee) a relationship between two files
- Reaches LRU performance when cache becomes larger

Cloud Storage – 2016

- S. Yang *et al.* Tombolo: Performance Enhancements for Cloud Storage Gateways
- Instead of file, probability graph on block
- Probability:
    - Consecutive block $P[N_n N_{n+1}] = \frac{1+\#[N_n N_{n+1}]}{1+\#[N_n]}$
    - Non-consecutive $P[N_n N_m] = \frac{\#[N_n N_m]}{\#[N_n]}$
- Select candidate blocks in a similar way to PageRank

Cloud Storage – 2016

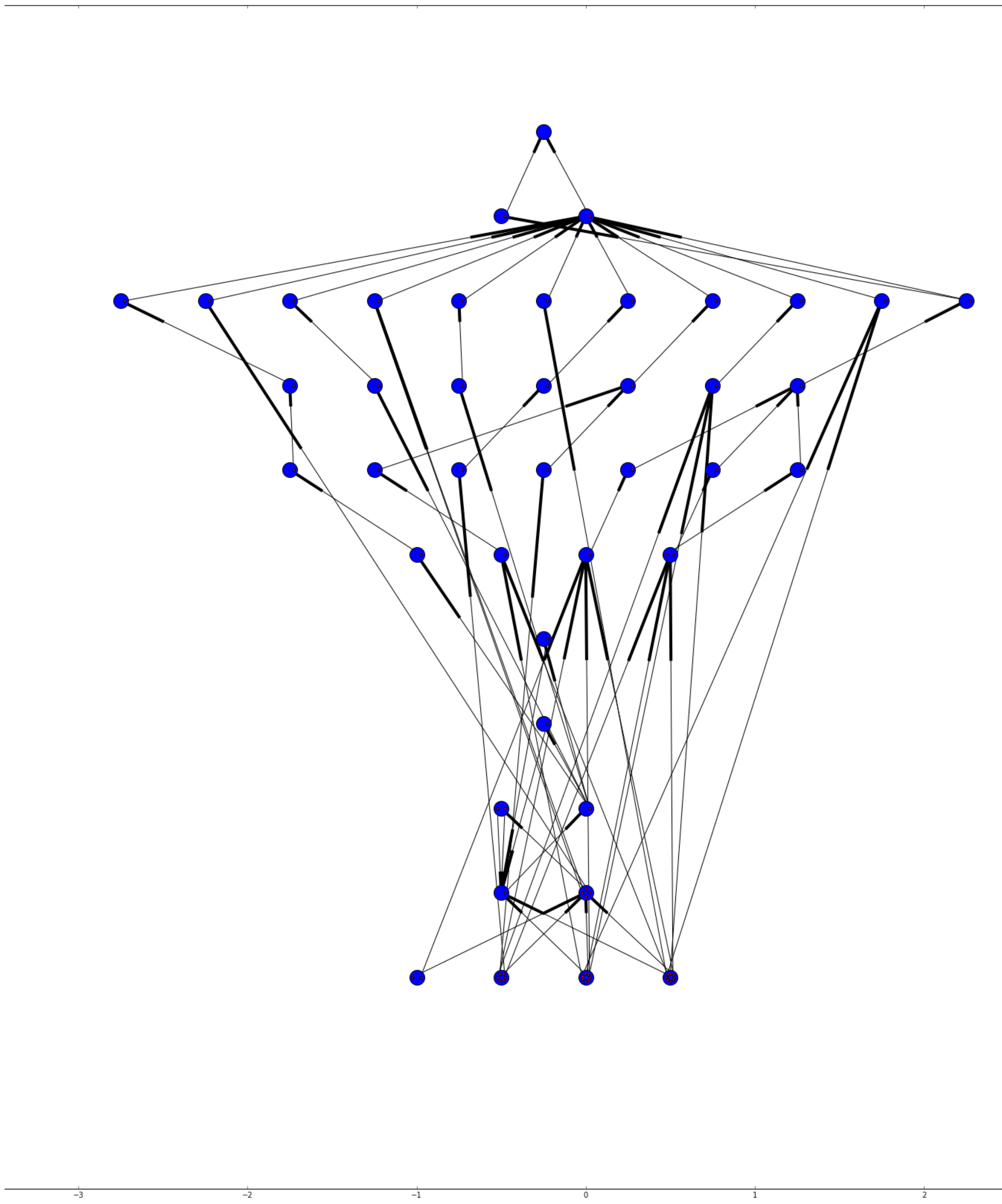


Can we do better?

- From files/blocks to processes (stack trace)
- Prefetching those never shown up in cache
    - Get the key of unseen files

Call Trace

- Example: phpBB
- Node = Function
- Gate = Line number



Next step

- Look at cache mechanism in Linux kernel