Experiments with Perfect Hashing
15-451
Feb. 20, 2001

- Perfect hashing
- Dictionaries
- Results

http://www.cs.cmu.edu/~bryant

Perfect Hashing Algorithm

Goal

- For key x in S, create unique hash signature (a_1,a_2)

Primary Hashing

- Hash x into M buckets to give a_1
- For those keys that hash into unique bucket, $a_2 = 0$

Secondary Hashing

- For each bucket containing k > 1 elements, create secondary table of size k^2
- Keep trying different hash functions h_1, h_2, …, until all elements in bucket hash to unique position in secondary table
- This gives value a_2

Dictionaries

Normal

- /usr/dict/words
- N = 45,402 English words
  - From “Aarhus” to “Zurich”
- 1–28 characters long
  - “antidisestablishmentarianism”

Big

- http://ftp.fu-berlin.de/misc/dictionaries/unix-format/american
- 869,145 “words”
  - From “A2A” to “ZzzzzZZZzzzzzzZzzzzzzz”
  - Meant for use in password cracking?
- 2–48 characters long
  - “Karntnerstrasse-Rotenturmstrasse”

Hash Function

- Key x = $c_1\ c_2\ \ldots\ c_{len(K)}$

Function

- $h(x) = \sum (a_i * c_i + b_i) \bmod M$
  - Computed over Z_p, where p = 16,777,199
  - a_i's, b_i's random numbers in Z_p
  - All sums & products computed modulo p

Universal Family

Hash functions h_1, h_2, …

- Generate 64 different a_j's and b_j's
- $h_j(x) = \sum (a_{i+j} * c_i + b_{i+j}) \bmod M$

Experiments

Does This Work?

How Many Secondary Hash Functions are Required

How Big Should M Be?

- M small ➔ small primary table, but more secondary tables
- M large ➔ large primary table, but fewer secondary tables
- Expect ideal to be some intermediate value

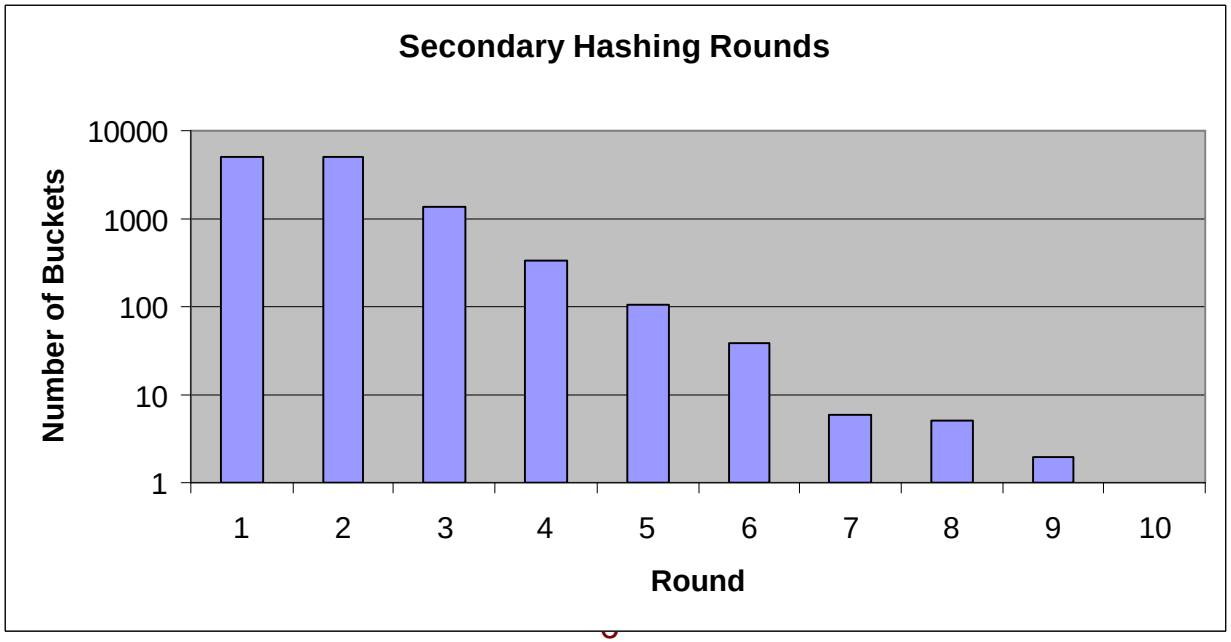
Hashing Normal Dictionary, M=N

Primary Hashing

- 16,694/45,402 (37%) have bucket size 1
- 11,968 buckets > 1
  - Biggest = 7

Secondary Hashing

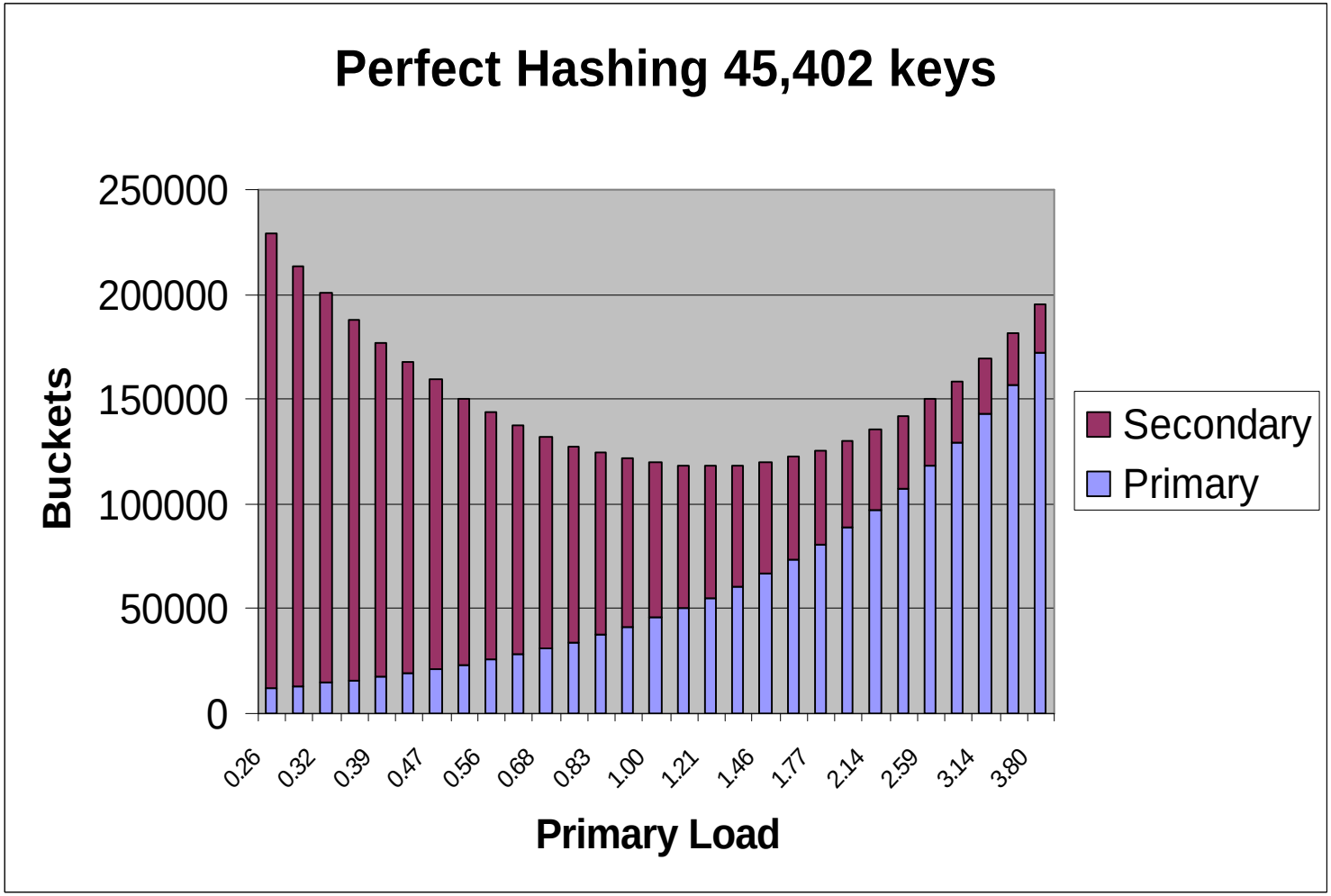
- 11,968 secondary tables, with total of 74,316 buckets
- Average #tries to find good hash function = 1.79



Normal Dictionary, Varying M/N

Total Number of Buckets

- 119,718 (2.63N) when M=N=45,402
- 117,759 (2.60N) when M=54,936



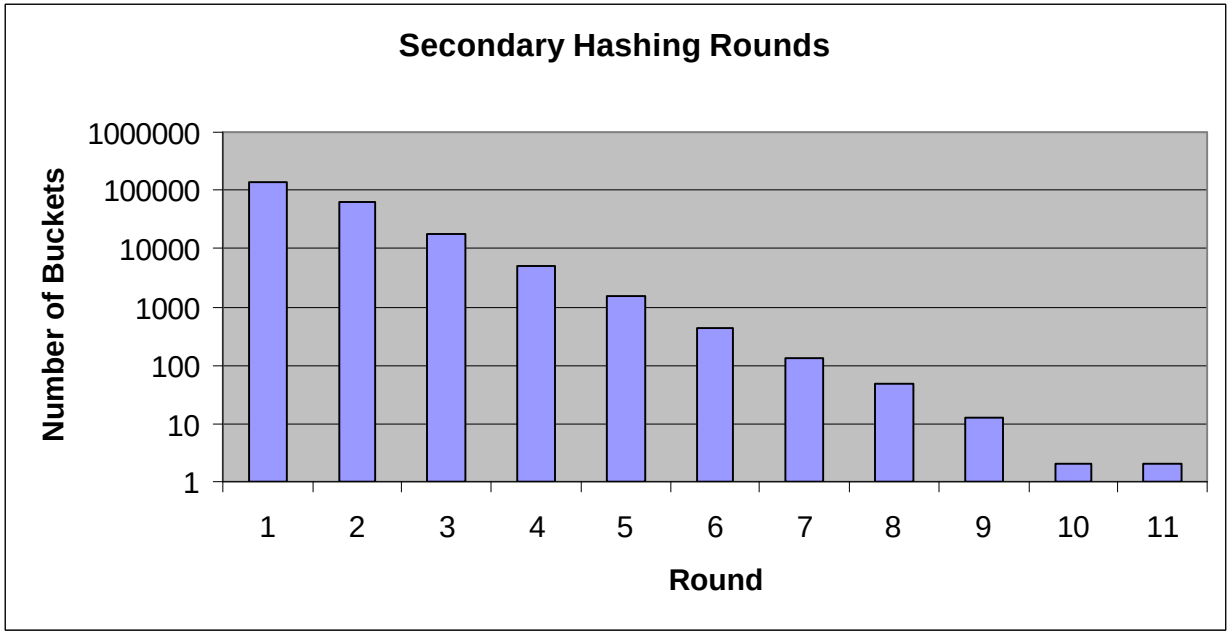
Hashing Big Dictionary, M=N

Primary Hashing

- 320,196/869,145 (37%) have bucket size 1
- 229,475 buckets > 1
  - Biggest = 10

Secondary Hashing

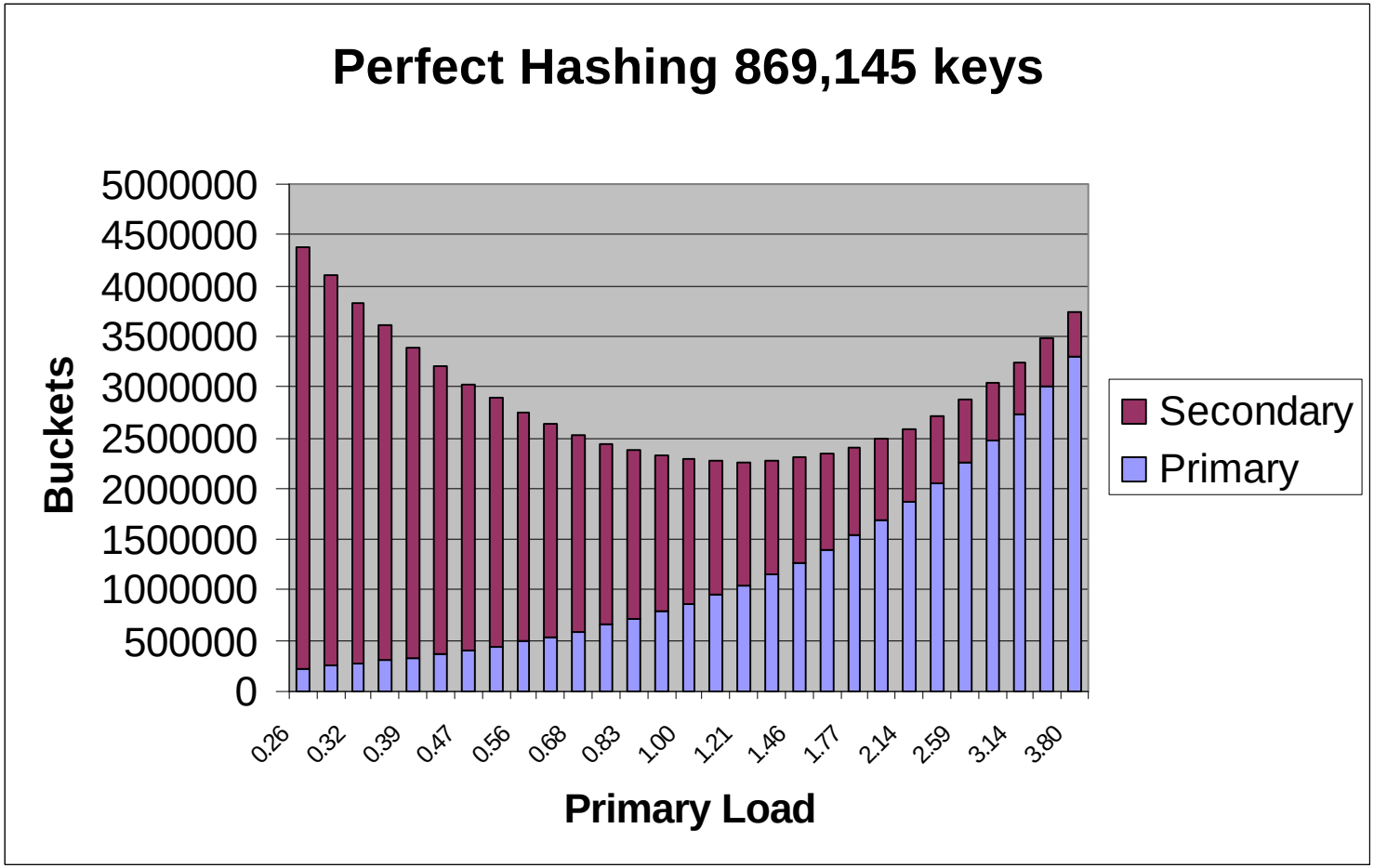
- 229,475 secondary tables, with total of 1,417,683 buckets
- Average #tries to find good hash function = 1.55



Normal Dictionary, Varying M/N

Total Number of Buckets

- 2,286,828 (2.63N) when M=N=869,145
- 2,256,093 (2.60N) when M=1,051,666



Observations

Does This Work?

- Yes. Theory is good predictor of reality
- Total of ~2.63N buckets
  - (3–1/e) N

How Many Secondary Hash Functions are Required

- Maximum of 13
- Average # tries to get good hash < 2

How Big Should M Be?

- M = N turns out to be nearly optimal