Content-based Page Sharing with Universal and Perfect Hashing

Xiaohai Yu
Dec. 11, 2009
In recent years, the speed and capacity gap between processor and memory continues to widen.

Methods for efficient usage of space resource are becoming increasingly important.

Rather than pursuing extreme time efficiency as the previous decade.
Multiple Virtual Machines

- **Case:** running multiple virtual machine instances on one single workstation
- **Want:** share memory across these VM instances to improve the memory usage efficiency.
- **Observe:** many pages are identical across the VMs, such as system kernel, device drivers, TCP/IP stack and etc.
VMware ESX Server

- VMware ESX Server allocates one single memory space for each instance.
- Each memory page in such spaces has its own PPN (Physical Page Number).
- VMware ESX Server maps these pages into the memory of the host machine, which maps the PPN to MPN (Machine Page Number).
Content-based page sharing

VM 1  VM 2  VM 3

Machine Memory

PPN 2868
MPN 1096

hash contents

...2bd806af

hint frame

hash: ...06af
MPN: 123b
VM: 3
PPN: 43f8

shared frame

hash: ...07d8
MPN: 8f44
refs: 4
Key problem: Hash function

- **Key problem**: to construct a hash function mapping the length $L$ long identifier to hash value of $O(\log L)$ length.
- Modern memory pages are usually 4K to 512K bits, which is a very long input for the hash function.
- Moreover we want the description length of this hash function should also be $O(\log L)$. 
Hash Basics

- A universe $U$ with some subset $N \in U$. We want to store the subset $N$ using as little space as possible.
- This function $h: U \rightarrow \{1,2,\ldots,M\}$ is defined as the hash function.

**Definition:** A collision occurs when $h(x) = h(y)$ for two distinct keys $x, y$. 
Hash function

- **Claim:** Let $F$ be a hash function, that maps $n$ elements to table $[m]$, with proper $m$, then the expected number of collisions will be at most $\frac{1}{2}$

- **Proof:**

\[
Pr[F \text{ is not } 1-1] = Pr[\exists i \neq j: F(E_i) = F(E_j)] \\
= Pr[F(E_1) \neq F(E_2) \text{ or } F(E_2) \neq F(E_3) \text{ or } \ldots] \\
\leq \left(\frac{n}{2}\right)Pr[F(E_1) \neq F(E_2)] \\
= \left(\frac{n}{2}\right) \frac{1}{m} \\
= \frac{n(n-1)}{2} \frac{1}{n^2} \leq \frac{1}{2}
\]
Universal Hashing

- **A universal hash function** is one in which the probability of a collision between any two keys is provably $1/M$.

- **Definition**: A randomized algorithm $H$ for constructing hash function $h: U \to \{1,2,\ldots, M\}$ is **universal** if for all $x<>y$ in $U$, we have

  $$\Pr[h(x) = h(y)] \leq \frac{1}{M}$$
Recall: Finite Fields

- A finite field \( F \) is a set of objects with operations + and * that behave as you would expect as real space.

- **Example**: In a field \( F=\{0,1,2,\ldots,12\} \) with operation +, * and mod 13:

\[
\begin{align*}
12 + 2 &= 1 \\
3 * 5 &= 3 \\
1/2 &= 7
\end{align*}
\]

- **Observe**: For every prime \( P \) the above field with mod \( P \) is a finite field.
Polynomial over Finite Fields

- A polynomial over finite fields is an expression of the form
  \[ \sum_{i=1}^{N} (a_i \times x^{i-1}) \mod p \]

- For some non-negative integer \( n \) and where the coefficients are drawn from some designated set \( S \).
- \( S \) is called the coefficient set. When \( a \neq 0 \), we have a polynomial of degree \( n \).
Polynomial arithmetic

- We can add two polynomials:

  \[ f(x) = a_2x^2 + a_1x + a_0 \mod P \]
  \[ g(x) = b_1x + b_0 \mod P \]
  \[ f(x) + g(x) = a_2x^2 + (a_1 + b_1)x + (a_0 + b_0) \mod P \]

- We can multiply two polynomials:

  \[ f(x) = a_2x^2 + a_1x + a_0 \mod P \]
  \[ g(x) = b_1x + b_0 \mod P \]
  \[ f(x) \times g(x) = a_2b_1x^3 + (a_2b_0 + a_1b_1)x^2 + (a_1b_0 + a_0b_1)x + a_0b_0 \mod P \]

- **Theorem**: Non-zero polynomial \( P(x) \) over a finite field \( F \), with degree \( d \), has at most \( d \) roots.
Hash function for long identifiers

- Find a prime P slightly larger than $L^2$
- Using modular P to define a hash function like following:

$$H_x(ID) \equiv P_{ID}(x) = \sum_{i=1}^{N} (ID[i] \times x^{i-1}) \mod P$$

- Where ID[i] denotes the i-th bit of long identifier ID, and x is picked at random in [1..P].
- Represent the identifier as a polynomial over a finite field with modular P.
Proof: Universal Hashing

• **Claim**: For any two distinct IDs, the probability, over the choice of the hash function that their hashes coincide is at most $1/L$.

• Approximately consider that this hash function hashes $L$-bit ID to a number in $[1..L^2]$, which is of length $2\times\log(L)$.
Proof (cont’)

- For any two different ID and ID’, both of length L, will show that the probability that $=$ can be induced to the probability taken over random x on. The equation $*$:

$$H_x(ID) - H_x(ID') = \sum_{i=1}^{L} ((ID[i] - ID'[i]) \times x^{i-1}) \mod P$$

- Assume ID and ID’ have s bits common from the beginning, it will be a polynomial mod P of degree L-1-s.
Proof (cont’)

- So $Pr[H_x(ID) - H_x(ID') = 0]$ will be the number of x’s roots in the equation * over the size of original ID set.
- (Note: P is a prime and we showed polynomial over a finite field with operation modular P has most degree roots)

$$Pr[H_x(ID) - H_x(ID') = 0] = \frac{L-1-s}{p} < \frac{L-1}{L^2} < \frac{L}{L^2} = \frac{1}{L}$$

- Thus we prove the collision is at most 1/L.
Perfect Hashing

- The above construction is the first step of my hash function construction. What we want next is Perfect Hashing, with zero collision.
- Idea: two level hashing. For all the buckets with collision, we generate another hash function which will give no collisions for the items in such buckets.
- Still bound the total space of the hash function.
Future Work
Thank You!