Map Reduce : Input (k-v), map (병렬가능), reduce (병렬불가)

문제 상황 : Top 10 url 찾고 count 해라

process : Input k-v → 최초 k-v → Groupby → output k-v

문제상황 2 : 각 host가 접속하는 url의 총 용량 ⇒ JOIN 으로 해결

⇒ hash function 사용

Map 에서 나온 값들을 중간 file에 저장하여 Reduce 와 송수신

구성 : Master node, map node, reduce node

↳ 노드관리, Helath check

if, map 죽으면 다른 worker, reduce 죽어도 재시작

master 죽으면 관리자 알림

Backup Task : 처리량 너무많아서 느려진경우 → 가까운애들에 할당

- Combiners : 중간 결과 저장 및 Reducer 에 전송
- Partition function : Hash 통해 key 비교

Fixed propotion : 비율고정, uniform distribution 가정,

문제상황 : 한 user가 한 달에 같은 쿼리를 얼마나 검색 하는지

해결 : user sampling (hash 이용)

Fixed-size tuple : tuple size 고정

Problem : 새로운 데이터 들어올때 뽑힐 확률이 다 달라진다.

Solution : 저장 공간 S보다 time step n이 크면 $\frac{S}{n}$ 확률로 다음 item을 선택여부 결정

∴ $\frac{S}{n}$ 의 확률로 데이터가 뽑힐 확률 유지

Sliding Window : 가장 최근 N 개만 본다

문제상황 : 0과 1이 들어올때 최근 k bit 에 대해 1의개수는? (단, window size k bit 작용)

↳ Solution Method ① Exponential Windows

마지막 사각형 전까진 1의개수 완벽하게 측정가능, but 나머진 측정 (최대 50% 에러)

사각형 안엔 2^k 개의 1의개수 저장공간 $O(\log^2 N)$ 1의 균등하게 분배된다는 가정

DGIM Method

⇒ window size 가변적 (1의 개수에 따라 window size 변화

Timestamps ⇒ 위치에 대한 정보, N에 비례 $O(\log_2 N)$ bit 필요

Buckets ⇒ end 지점, 1의개수 저장 $O(\log N)$ bit

2^n 저장이 아니라 n 만 저장 ∴ $O(\log \log N)$

제약조건 : 1개 or 2개 같은사이즈 bucket 존재, overlap X, t↑→ size↓, end_time > N 일때 사라짐

update : 0이 들어오면 무시, 1 들어오면 새로운 bucket 생성

1개씩 세는 법 : 마지막 bucket 빼고 모두 더함. 마지막 bucket의 반만 더함

에러율 : 윈도우 개수 r 개 일때 Error $O(\frac{1}{r})$

Bloom Filter : spam msg 판단

특징 : FP(긍정예측, 사실부정) 은 있지만 FN 없다.

method : hash function 이용하여 Bit 값 채움

ex) $h_1(x) = x \% 11$ (odd), $h_2(x) = x \% 11$ (even) ← x = binary, 오른쪽부터 읽을

input = 159 = 10011111 짝 : 0111 % 11 = 7 홀 : 1011 % 11 = 0 10000001000

B[h(a)] = 1 ⇒ setting process

118 = 1110110 $h_1(y) = 3$, $h_2(y) = 5$ 둘다 0 ∴ 리스트 안에 없다 = Spam

Anaylsis : Hash function 하나일 때 Bit에 1이 채워질 확률

$= 1 - e^{-\frac{m}{n}} \approx$ FP 확률 m = Input 아이템, n = Target = bits = bucket

if hash function ⇑ → 겹칠도 ⇑ → 1이들어갈확률 $1 - e^{\frac{km}{n}}$

$FP = (1 - e^{-\frac{km}{n}})^k$ ∴ FP ⇓

Counting Distinct Elements

input x → hash → binary (ex. 1100) ⇒ r(a) = 2 (오른쪽부터 0의 개수)

max(r(a)) = R 2^R = Unique value counts

$e^{-m2^{-r}}$ → $2^R \approx m$

Counting Moments : $\sum (m_i)^k$ ← 적률

k = 2 이면 얼마나 분포가 불균등한지 나타냄 S⇑ → uneven

AMS Method : 기대값으로 추정

⇒ $n(2c-1)$ ∴ $n(c^k - (c-1)^k)$ n = stream 개수, c = count

⇒ 실제로 n을 모르기때문에 factor 로 둔다.

Counting items ⇒ Exponentially Decaying Windows

⇒ element 당 점수 매긴다. $\sum_{i=1}^{t} \delta(1-c)^{t-i}$ ⇒ 데이터가 새로들어올수록 먼저들어온 데이터에 대한 중요도 ⇓, 새롭게 값이 들어오면 +1

⇒ 가장 많이 들어온 것은 weight 가 가장 큰것. sum(weight) = $\frac{1}{c}$

element 개수 < $\frac{2}{c}$

Bayes = $P(x|y) = \frac{P(x)P(y|x)}{P(y)}$

chain rule = $P(a,b,c) = P(A|B \cap C) \times P(B|C) \times P(C)$

Entropy = $H(x) = -p \log p - (1-p) \log (1-p)$

Cross Entropy = $H(P) + D_{KL}(P||Q)$

$= H(P) + E_{x \sim p}[\log P(x) - \log Q(x)]$