CSE544
Data Management

Lectures 14

Parallel Query Processing

Skew

Skew

- Skew in the input: a data value has much higher frequency than others
- Skew in the output: a server generates many more values than others, e.g. join
- Skew in the computation

Some Skew Handling Techniques

If using range partition:

- Ensure each range gets same number of tuples
- E.g.: {1, 1, 1, 2, 3, 4, 5, 6 } → [1,2] and [3,6]
- Eq-depth v.s. eq-width histograms

Some Skew Handling Techniques

Create more partitions than nodes

- And be smart about scheduling the partitions

- Note: MapReduce uses this technique
  – We will talk about MapReduce later

Input Skew

- We will discuss how to manage skew in the input
- Recall *Skew join*: partition values into light and heavy, join them separately
- Need to define *light* and *heavy*

Problem Statement

Given: N data items $x_1, \ldots, x_N$

- We hash-partition them to P nodes
- When is the partitioning uniform?

Problem Statement

Given: N data items $x_1, \ldots, x_N$

- We hash-partition them to P nodes
- When is the partitioning uniform?

Uniform: each node has O(N/P) items

Problem Statement

Given: N data items $x_1, \ldots, x_N$

- We hash-partition them to P nodes
- When is the partitioning uniform?

Uniform: each node has O(N/P) items

Skew: some node has >> N/P items

Problem Statement

Given: N data items $x_1, \ldots, x_N$

- We hash-partition them to P nodes
- When is the partitioning uniform?

Uniform: each node has O(N/P) items

Skew: some node has >> N/P items

1. Because of the hash function h, or
2. Because of skew in the data

1. Role of the Hash Function

Assume $x_1, \ldots, x_N$ are distinct

Hash function computes $h(x_i) \in \{1, \ldots, P\}$

- If h is *fixed* then we can find bad items that will overload one server; how?

1. Role of the Hash Function

Assume $x_1, \ldots, x_N$ are distinct

Hash function computes $h(x_i) \in \{1,\ldots,P\}$

- If h is *fixed* then we can find bad items that will overload one server; how?
- If h is *random* (every day we choose another h…): *balls-in-bins* problem

1. Role of the Hash Function

Fix a node j ∈ {1,…,P}; compute its load

1. Role of the Hash Function

Fix a node $j \in \{1,\ldots,P\}$; compute its load

- $\forall x_i$, $\Pr(h(x_i) = j) = 1/P$

1. Role of the Hash Function

Fix a node $j \in \{1,\ldots,P\}$; compute its load

- $\forall x_i$, $\Pr(h(x_i) = j) = 1/P$
- $E[\ \mathrm{Load}(j)\ ] = \sum_{i=1,N} \Pr(h(x_i) = j) = N/P$

1. Role of the Hash Function

Fix a node $j \in \{1,\ldots,P\}$; compute its load

- $\forall x_i$, $\Pr(h(x_i) = j) = 1/P$
- $E[\text{ Load}(j)] = \sum_{i=1,N} \Pr(h(x_i) = j) = N/P$
- Fix $\delta \in (0,1]$; call the node j "bad" if

$$\text{Load}(j) > (1+ \delta)\ N/P$$

1. Role of the Hash Function

Fix a node $j \in \{1,\ldots,P\}$; compute its load

- $\forall x_i$, $\Pr(\ h(x_i) = j) = 1/P$
- $E[\ \text{Load}(j)\ ] = \sum_{i=1,N} \Pr(\ h(x_i) = j) = N/P$
- Fix $\delta \in (0,1]$; call the node j "bad" if

  $$\text{Load}(j) > (1+\delta)\ N/P$$
- Chernoff Bound:

  $$\Pr(\ \text{bad}(j)\ ) \le \exp(-\ \delta^2/3 * N/P)$$

1. Role of the Hash Function

Fix a node $j \in \{1,\ldots,P\}$; compute its load

- $\forall x_i$, $\Pr(h(x_i) = j) = 1/P$
- $E[\ \mathrm{Load}(j)\ ] = \sum_{i=1,N} \Pr(h(x_i) = j) = N/P$
- Fix $\delta \in (0,1]$; call the node j "bad" if

  $$\mathrm{Load}(j) > (1+ \delta)\ N/P$$

- Chernoff Bound:

  $$\Pr(\ \mathrm{bad}(j)\ ) \leq \exp(- \delta^2/3 * N/P)$$

$\Pr(\ \mathrm{bad}(1) \vee \ldots \vee \mathrm{bad}(P)\ ) \leq P * \exp(-\delta^2/3 * N/P)$

1. Role of the Hash Function

Summary: Bad(j) = (Load(j) > (1+ δ) N/P)

Pr(bad(1)∨…∨bad(P)) ≤ P *exp(-δ^2/3*N/P)

1. Role of the Hash Function

Summary: Bad(j) = (Load(j) > (1+ δ) N/P)

Pr(bad(1)∨…∨bad(P)) ≤ P *exp(-δ^2/3*N/P)

- When N/P is large, then the probability of having any "bad" node is very small;

1. Role of the Hash Function

Summary: $\text{Bad}(j) = (\text{Load}(j) > (1+\delta)\, N/P)$

$\Pr(\text{bad}(1) \vee \ldots \vee \text{bad}(P)) \leq P \cdot \exp(-\delta^2/3 \cdot N/P)$

- When N/P is large, then the probability of having any "bad" node is very small;
  E.g. $\delta = 0.5$, $N > P \cdot \log(P)$ then:

  Pr(some node has load>150%) < $1/P^{0.92}$

1. Role of the Hash Function

Summary: Bad(j) = (Load(j) > (1+ δ) N/P)

Pr(bad(1)∨…∨bad(P)) ≤ P *exp(-δ^2/3*N/P)

- When N/P is large, then the probability of having any "bad" node is very small;
  E.g. δ = 0.5, N > P*log(P) then:

  Pr(some node has load>150%) < $1/P^{0.92}$

- When N=P then this argument won't work;
  Balls in bins: E[Load(j)] = 1 but

  E[$\max_j$ Load(j)] ≈2 log N/log(log(N))

1. Role of the Hash Function

Takeaways

- Don’t write your own hash function
- Randomize it (how?)
- Make sure you have enough items, N >> P (otherwise, why parallelize?)

Then Load = O(N/P)

2. Role of the Data Skew

Assume $x_1, \ldots, x_N$ may have duplicates

2. Role of the Data Skew

Assume $x_1, \ldots, x_N$ may have duplicates

- Fact: if some item x_i occurs > N/P times then some node j has: Load(j) > N/P

2. Role of the Data Skew

Assume x_1, …, x_N may have duplicates

- Fact: if some item x_i occurs > N/P times then some node j has: Load(j) > N/P

Conversely, if every value occurs << N/P times, then load per node is O(N/P)

2. Role of the Data Skew

Assume $x_1, \ldots, x_N$ may have duplicates

- Fact: if some item x_i occurs > N/P times then some node j has: Load(j) > N/P

Conversely, if every value occurs << N/P times, then load per node is O(N/P)

Proof sketch: assume each item occurs exactly K times. Thus N/K distinct items, and we use previous argument

Light and Heavy Hitters

- Light hitters: values that occur << N/P times
  - For the precise threshold use some fudge factor that is about log(P)

- Heavy hitters: the others
  - Good news: there are only about $\tilde{O}(P)$ heavy hitter values. (Why?)

Skew Join - Recap

$R(A,B) \bowtie_{B=C} S(C,D)$

- Problem: skewed values C in S
- Preprocessing: identify heavy hitter values C (occur > N/P / fudge-factor)
- Partition S into S^{light} and S^{heavy}
- Use partition hash-join for $R \bowtie S^{light}$
- Use broadcast join for $R \bowtie S^{heavy}$