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\} \rightarrow [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 $O\left(\frac{N}{P}\right)$ 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 \( \gg N/P \) 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 $O\left(\frac{N}{P}\right)$ items
  Skew: some node has $>> \frac{N}{P}$ 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 \textit{fixed} then we can find bad items that will overload one server; \textit{how}?

- If $h$ is \textit{random} (every day we choose another $h$...): \textit{balls-in-bins} problem
1. Role of the Hash Function

Fix a node $j \in \{1, \ldots, P\}$; compute its load
1. Role of the Hash Function

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

- $\forall x_i$, $\Pr(h(x_i) = j) = \frac{1}{P}$
1. Role of the Hash Function

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

- $\forall x_i$, $\Pr( h(x_i) = j) = 1/P$
- $\mathbb{E}[\text{Load}(j)] = \sum_{i=1}^{N} \Pr( h(x_i) = j) = N/P$
1. Role of the Hash Function

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

- $\forall x_i$, $\Pr( h(x_i) = j ) = 1/P$
- $\mathbb{E}[ \text{Load}(j) ] = \sum_{i=1}^{N} \Pr( h(x_i) = j ) = N/P$
- 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:
  $\Pr( \text{bad}(j) ) \leq \exp(-\delta^2/3 * N/P)$
1. Role of the Hash Function

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

- \( \forall x_i, \ Pr( h(x_i) = j) = 1/P \)
- \( E[ \text{Load}(j) ] = \sum_{i=1,N} Pr( h(x_i) = j) = N/P \)
- Fix \( \delta \in (0,1] \); call the node \( j \) “bad” if \( \text{Load}(j) > (1+\delta) N/P \)
- Chernoff Bound:
  \[
  Pr( \text{bad}(j) ) \leq \exp(-\delta^2/3 * N/P)
  \]
  \[
  Pr( \text{bad}(1) \lor \ldots \lor \text{bad}(P) ) \leq P \cdot \exp(-\delta^2/3*N/P)
  \]
1. Role of the Hash Function

Summary: \[
\text{Bad}(j) = (\text{Load}(j) > (1+ \delta) \frac{N}{P})
\]
\[
\Pr(\text{bad}(1) \vee \ldots \vee \text{bad}(P)) \leq P \exp(-\frac{\delta^2}{3} \frac{N}{P})
\]

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

• When \(N=P\) then this argument won't work;

Balls in bins: \(E[\text{Load}(j)] = 1\) but
1. Role of the Hash Function

Summary:  \[ \text{Bad}(j) = (\text{Load}(j) > (1+ \delta) \frac{N}{P}) \]
\[ \Pr(\text{bad}(1) \lor \ldots \lor \text{bad}(P)) \leq P \times \exp(-\frac{\delta^2}{3P}) \]

• When \( N/P \) is large, then the probability of having any “bad” node is very small;
1. Role of the Hash Function

Summary: \[ \text{Bad}(j) = (\text{Load}(j) > (1+\delta) \frac{N}{P}) \]
\[ \Pr(\text{bad}(1) \lor \ldots \lor \text{bad}(P)) \leq P \exp(-\delta^2/3 \times \frac{N}{P}) \]

- When \( N/P \) is large, then the probability of having any “bad” node is very small;
  E.g. \( \delta = 0.5, N > P \log(P) \) then:
  \[ \Pr(\text{some node has load}>150%) < \frac{1}{P^{0.92}} \]
1. Role of the Hash Function

Summary: \[ \text{Bad}(j) = (\text{Load}(j) > (1+\delta) \frac{N}{P}) \]
\[ \Pr(\text{bad}(1) \lor \ldots \lor \text{bad}(P)) \leq P \times \exp(-\delta^2/3 \times \frac{N}{P}) \]

- When \( \frac{N}{P} \) is large, then the probability of having any “bad” node is very small;
  E.g. \( \delta = 0.5, \ N > P \times \log(P) \) then:
  \[ \Pr(\text{some node has load}>150\%) < \frac{1}{P^{0.92}} \]

- When \( N=P \) then this argument won’t work;
  Balls in bins: \( E[\text{Load}(j)] = 1 \) but
  \[ E[\max_j \text{Load}(j)] \approx 2 \log \frac{N}{\log(\log(N))} \]
1. Role of the Hash Function

Takeaways

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

Then Load = $O(N/P)$
2. Role of the Data Skew

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

• Fact: if $x_1 = x_2 = \ldots = x_N$ then some node $j$ has: $\text{Load}(j) = N$

• Fact: if some item $x_i$ occurs $> N/P$ times then some node $j$ has: $\text{Load}(j) > N/P$

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 $> \frac{N}{P}$ times, then some node $j$ has: $\text{Load}(j) > \frac{N}{P}$
2. Role of the Data Skew

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

- Fact: if some item $x_i$ occurs $> \frac{N}{P}$ times then some node $j$ has: $\text{Load}(j) > \frac{N}{P}$

Conversely, if every value occurs $<< \frac{N}{P}$ times, then load per node is $O\left(\frac{N}{P}\right)$
2. Role of the Data Skew

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

- Fact: if some item $x_i$ occurs $> \frac{N}{P}$ times then some node $j$ has: $\text{Load}(j) > \frac{N}{P}$

Conversely, if every value occurs $\ll \frac{N}{P}$ times, then load per node is $O\left(\frac{N}{P}\right)$

Proof sketch: assume each item occurs exactly $K$ times. Thus $\frac{N}{K}$ distinct items, and we use previous argument
Light and Heavy Hitters

• Light hitters: values that occur $\ll \frac{N}{P}$ 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 \( S^{\text{light}} \) and \( S^{\text{heavy}} \)
- Use partition hash-join for \( R \bowtie S^{\text{light}} \)
- Use broadcast join for \( R \bowtie S^{\text{heavy}} \)