Lecture 16

Tuesday, 29 October 2019 14:34

Parallel join algorithms

Join $q(x, y, z): -R(x, y), S(y, z)$ where R has N_R tuples and S has N_S tuples
Parallel hash join: partition both partitions per join variable. Requires no skew in the hash function for load $\frac{N_R}{p} + \frac{N_S}{p} = O\left(\frac{N}{p}\right)$
Broadcast join: broadcast smallest relation, partition the larger using both attributes. Load $N_R + \frac{N_S}{p}$.
Requires $N_R \ll N_S$ for $O\left(\frac{N}{p}\right)$
Cartesian product: general (applies to θ joins). Reshape the p machines into $p_x \times p_z$.
Hash the values of x and y into $\{1, \dots, p_x\}$ and $\{1, \dots, p_y\}$ respectively; distribute to machine (p_i, p_j) the tuples hashed with (p_i, p_j). Load $\frac{N_R}{p_x} + \frac{N_S}{p_z}$. Then $p_x = \sqrt{p\frac{N_R}{N_S}}, p_z = \sqrt{p\frac{N_S}{N_R}}$, thus load is $2\sqrt{\frac{N_R N_S}{p}}$. If $N_R = N_S$, then load is $\frac{N}{\sqrt{p}}$. In the limit where $N_R \ll N_S$, then $p_x = 1$, and we achieve the broadcast join algorithm. Therefore the load is $O\left(\max\left\{\frac{N_R}{p}, \frac{N_S}{p}, \sqrt{\frac{N_R N_S}{p}}\right\}\right)$

Generalises to multiple dimensions. Can be used for cyclic queries, e.g. Triangle query:
$\Delta(x, y, z): -R(x, y), S(y, z), T(z, x)$

$$L = \frac{N_R}{p_x p_y} + \frac{N_S}{p_y p_z} + \frac{N_T}{p_x p_z} = O\left(\frac{N}{p^{\frac{2}{3}}}\right)$$

Join algorithms with skew

Heavy hitter: value of a join variable y that appears more than $\frac{N}{p}$ times
Skew join algorithm:
For light hitter values of y: use parallel hash join algorithm
For every heavy hitter value a of y with frequencies $d_{a,R}$, $d_{a,S}$: use Cartesian product algorithm with p_a machines, where $\sum_a p_a = p$ and p_a is proportional to $d_{a,R}, d_{a,S}$

$$L = \max\left\{\sqrt{\frac{\sum_a d_{a,R} d_{a,S}}{p}}, \frac{N_R}{p}, \frac{N_S}{p}\right\} = \max\left\{\sqrt{\frac{|R \bowtie S|}{p}}, \frac{N_R}{p}, \frac{N_S}{p}\right\}$$