Processing of Probabilistic Skyline Queries Using MapReduce

Qingfu Wen

School of Software, Tsinghua University

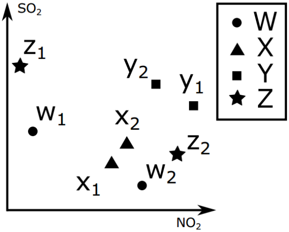
qingfu.wen@gmail.com

Author: Yoonjae Park, Jun-Ki Min, Kyuseok Shim

November 14, 2015

Probabilistic Skylines

Object	Instance	NO_2	SO_2	Probability
W	w_1	10	40	0.5
	w_2	75	10	0.4
X	x_1	55	20	0.2
	x_2	65	30	0.2
Y	y_1	95	60	0.8
	y_2	80	70	0.2
Z	z_1	5	80	0.5
	z_2	90	25	0.5



$$
\begin{aligned}
P_{sky}(y_1) &= P(y_1)(1 - P(w_1) - P(w_2))(1 - P(x_1) - P(x_2))(1 - P(z_2)) \\
&= 0.024 \\
P_{sky}(y_2) &= 0.012 \\
P_{sky}(Y) &= P_{sky}(y_1) + P_{sky}(y_2) = 0.036 \\
P_{sky}(W) &= 0.9 \\
P_{sky}(X) &= 0.4 \\
P_{sky}(Z) &= 0.74
\end{aligned}
$$

Probabilistic Skylines

Probabilistic Skyline Problem

For a set of uncertain objects $\mathbb{D}$ and a probability threshold T_p, the probabilistic skyline $pSL(\mathbb{D}, T_p)$, is the set of all objects whose skyline probabilities are at least T_p, $pSL(\mathbb{D}, T_p) = \{U \in \mathbb{D} | P_{sky}(U) \geq T_p\}$.

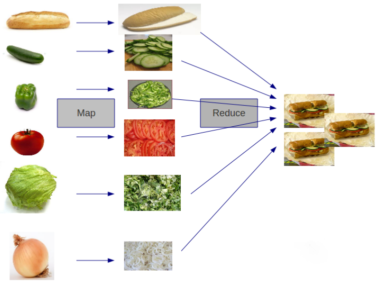
The discrete model:

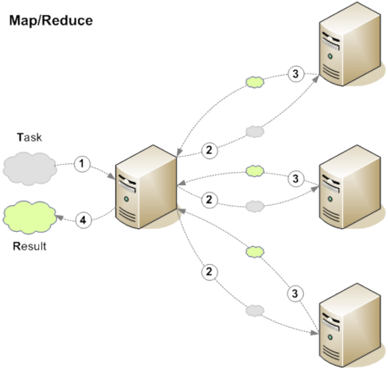
$$P_{sky}(U) = \sum_{u_i \in U} P_{sky}(u_i) = \sum_{u_i \in U} (P(u_i) \times \prod_{V \in \mathbb{D}, V \neq U} (1 - \sum_{v_j \in V, v_j \prec u_i} P(v_j)))$$

The continuous model:

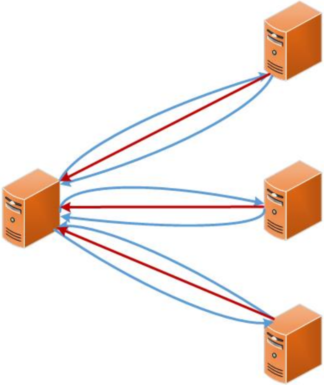
$$P_{sky}(u_i) = \int_U Uf(u) \times \prod_{V \in \mathbb{D}, V \neq U} (1 - \int_V Vf(v) 1(v \prec u)\, dv)\, du$$

What is MapReduce?





PSMR: The State-of-the-art Algorithm



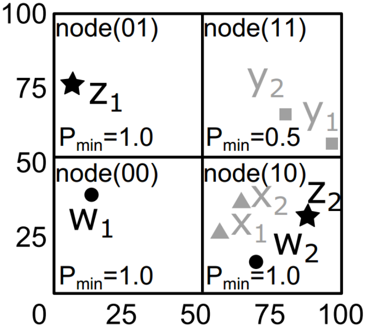
1. local computing candidate sets.
2. merge candidate sets, broadcast and local computing, reduce probabilities.

Early Pruning Techniques

Lemma (Zero-probability Filtering)

$$P_{sky}(U) = \sum_{u_i \in U} P(u_i) \times \prod_{V \in \mathbb{D}, V \neq U} \left(1 - \sum_{v_j \in V, v_j \prec u_i} P(v_j)\right)$$

delete u_i *if* $\prod_{V \in \mathbb{D}, V \neq U} \left(1 - \sum_{v_j \in V, v_j \prec u_i} P(v_j)\right) = 0.$



Example (Zero-probability Filtering)

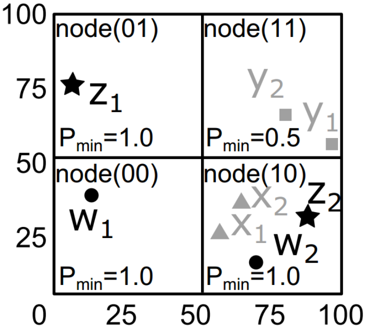
$$P_{sky}(y_1) = P(y_1)(1 - P(w_1) - P(w_2))(1 - P(x_1) - P(x_2))(1 - P(z_2))$$
$$\leq P(y_1)(1 - P(w_1)) = 0.4$$
$$P_{sky}(y_2) \leq 0.1$$
$$P_{sky}(Y) = P_{sky}(y_1) + P_{sky}(y_2) \leq 0.5 \leq T_p$$

Early Pruning Techniques

Lemma (Upper-bound Filtering)

$$\beta(U, \mathbb{S}, R(u_i)) = \frac{\prod_{V \in \mathbb{S}} (1 - \sum_{v_j \in V, v_j \prec R(u_i).min} P(v_j))}{1 - \sum_{v_k \in U, v_k \prec R(u_k).min} P(v_j)}$$

$$up(u_i, U, \mathbb{S}, R(u_i)) = P(u_i) \times \beta(U, \mathbb{S}, R(u_i)).$$



Example (Upper-bound Filtering)

$$\begin{aligned} P_{sky}(y_1) &= P(y_1)(1 - P(w_1) - P(w_2))(1 - P(x_1) - P(x_2))(1 - P(z_2)) \\ &\leq P(y_1)(1 - P(w_1)) = 0.4 \end{aligned}$$

$$P_{sky}(y_2) \leq 0.1$$

$$P_{sky}(Y) = P_{sky}(y_1) + P_{sky}(y_2) \leq 0.5 \leq T_p$$

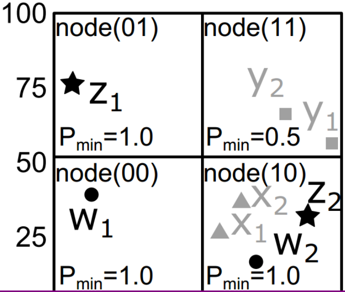
Early Pruning Techniques

Lemma (Dominance-Power Filtering)

$DP(v_j) = \prod_{1}^{d}(b(k) - v_j(k)) = 0, b(k) = max\{v_1(k), \cdots, v_n(k)\}.$

$DP(V) = \sum_{v_j \in V}(P(v_j) \times DP(v_j)).$

$\mathbb{F}$ is topK DP set, $\sum\limits_{u_i \in U} P(u_i) \times \prod\limits_{V \in \mathbb{F}, V \neq U} (1 - \sum\limits_{v_j \in V, v_j \prec u_i} P(v_j)) < T_p$, U is not a probabilistic skyline Object.



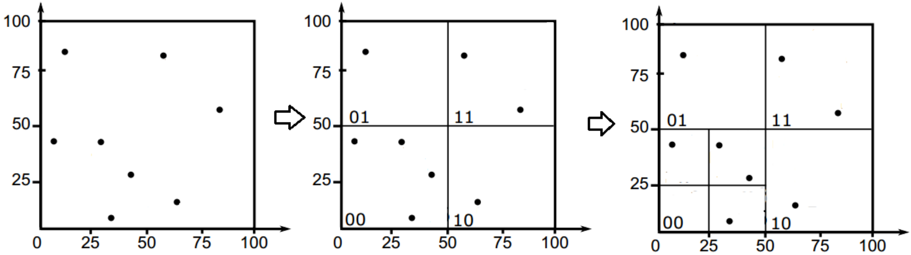
Example (Dominance-Power Filtering)

$$\begin{aligned} P_{sky}(y_1) &= P(y_1)(1 - P(w_1) - P(w_2))(1 - P(x_1) - P(x_2))(1 - P(z_2)) \\ &\leq P(y_1)(1 - P(w_1)) = 0.4 \end{aligned}$$

$$P_{sky}(y_2) \leq 0.1$$

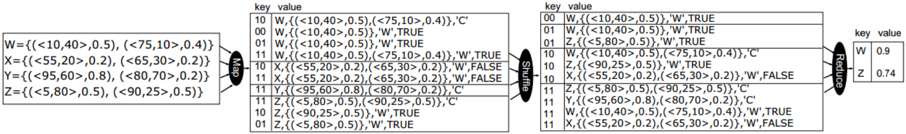
$$P_{sky}(Y) = P_{sky}(y_1) + P_{sky}(y_2) \leq 0.5 \leq T_p$$

PSQtree for Pruning



- generate PSQtree using a random sample $\mathbb{S}$ of $\mathbb{D}$
- traverse PSQtree for computing $P_{sky}(node.min)$
- zero-probability filtering
- upper-bound filtering
- partitioning objects by PSQtree

MapReduce Algorithms with PSQtree



Function PS-QPF-MR($\mathbb{D}$, T_p, ρ)
$\mathbb{D}$: uncertain dataset, T_p: probability threshold, ρ: split threshold
begin
$\mathbb{S}$ = Sample($\mathbb{D}$);
$PSQtree$ = GenQtree($\mathbb{S}$, ρ);
Broadcast $PSQtree$; Broadcast T_p;
pSL = RunMapReduce(PS-QPFC-MR, $\mathbb{D}$);
return pSL;
end

Function PS-QPFC-MR.setup()
begin
H = InitMinHeap(); $PSQtree$ = LoadPSQtree();
end

Function PS-QPFC-MR.map(U)
U: an uncertain object
begin
T_p = LoadThreshold();
U' = ZeroProb(U, $PSQtree$);
$upper$ = UpperBound(U', $PSQtree$);
$cand$ = **FALSE**;
if $upper \geq T_p$ **then**
  $cand$ = DP-Filter(U', T_p, H);
  if $cand$ **then** emit($n(U'.max)$, (U', 'C'));
for each leaf node n_ℓ in $PSQtree$ **do**
  if $cand$ = **True** and $n_\ell = n(U'.max)$ **then** continue;
  I = NewList();
  for each u_i in U' **do**
    if $n(u_i) \preceq n_\ell$ **then**
      I.add(u_i));
  emit(n_ℓ, (I, 'W', $cand$))
end

Function PS-QPFC-MR.reduce(n_ℓ, L)
begin
(L_C, L_W^T, L_W^F) = SplitList(L);
T_p = LoadThreshold();
for each object U in L_C **do**
  $skyline\_prob$ = SkylineProb(U, L_C, L_W^T, L_W^F);
  if $skyline\_prob \geq T_p$ **then**
    emit(U, $skyline\_prob$);
end

MapReduce Algorithms with PSQtree

- Reducing network overhead by clustering
- Workload balancing of reduce functions

Sample Size and Split Threshold of PSQtree

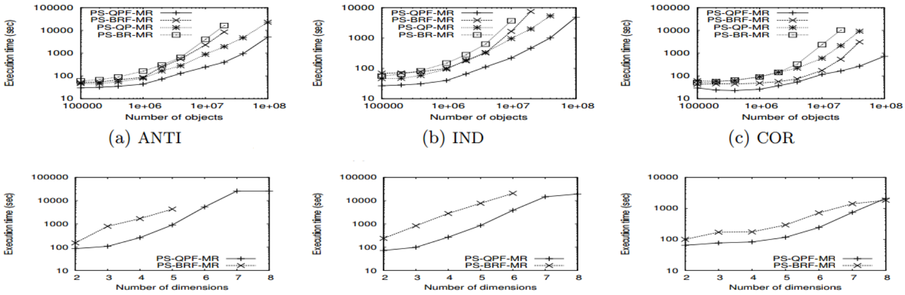
Experiments

- 50 machines with Intel i3 3.3GHz CPU and 4GB, Linux
- 200 machines with Intel Xeon 2.5GHz CPU and 3.75GB, Amazon EC2
- Java 1.6, Hadoop 1.2.1

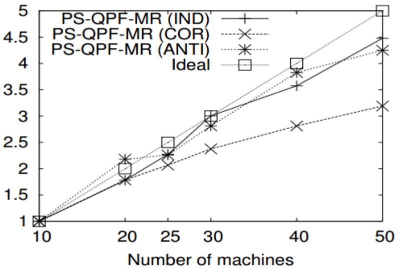
Algorithm	Description
PS-QP-MR	The algorithm with quadtree partitioning
PS-QPF-MR	The algorithm with quadtree partitioning and filtering
PS-BR-MR	The algorithm with random partitioning
PS-BRF-MR	The algorithm with random partitioning and filtering
PSMR	The state-of-the-art algorithm

Parameter	Range	Default
No. of samples($\mathbb{S}$)	1000∼10,000	1000 for PS-QPF-MR 2000 for PS-QP-MR 10000 for PS-BRF-MR
No. of dominating objects($\mathbb{F}$)	1000∼10,000	100 for PS-QPF-MR 1000 for PS-BRF-MR
No. of objects($\mathbb{D}$)	$10^5 \sim 10^8$	10^7
No. of dimensions(d)	$2 \sim 8$	4
Probability threshold (T_p)	0.1∼0.6	0.3
No. of inst. per object(ℓ)	$1 \sim 400$	40
No. of machines (t)	10∼200	25

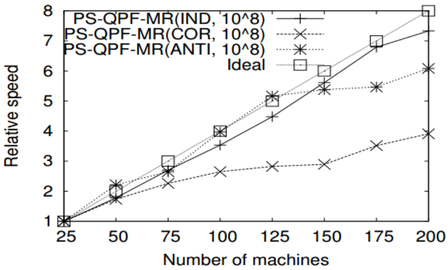
Experiments



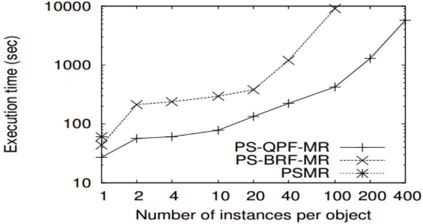
Experiments

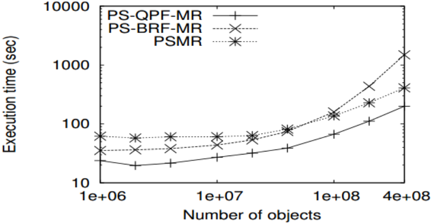


(a) With our cluster



(b) With Amazon EC2





Conclusion

- probabilistic skyline query for both discrete and continuous models
- zero-probability, the upper-bound, and dominance power filtering techniques
- using a PSQtree to distribute the instances of objects effectively
- a single MapReduce phase algorithm PS-QPF-MR and grouping optimization

The End

Q & A