

# Approximating Aggregation Queries in Peer-to-Peer Networks

Benjamin Arai – ([barai@cs.ucr.edu](mailto:barai@cs.ucr.edu))  
 Gautam Das – ([gdas@cse.uta.edu](mailto:gdas@cse.uta.edu))  
 Dimitrios Gunopulos – ([dg@cs.ucr.edu](mailto:dg@cs.ucr.edu))  
 Vana Kalogeraki – ([vana@cs.ucr.edu](mailto:vana@cs.ucr.edu))

<http://www.cs.ucr.edu/~barai>

# Peer-to-Peer Databases

• **P2P Database:** consists of numerous peer nodes that share data and resources with other peers on an equal basis

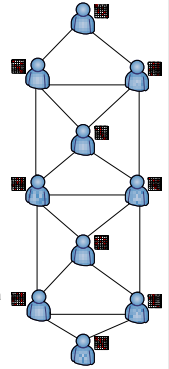
- No central coordination
- No central point of failure
- Highly Scalable
- Fault Tolerance
- Dynamic, nodes leave and enter the network with ease

• Applications of P2P Databases

- File sharing and retrieval (Gnutella & Freenet)
- VOIP Voice communication (Skype)

• Structured & Unstructured Topologies

- **Structured** networks are organized in such a way that data is organized in specific nodes with state information
- **Unstructured** networks assume no information about the location of data in the nodes where peers enter/leave without a priori notification



# Aggregation Queries

- Aggregation queries have the potential of finding applications in decision support, data analysis and data mining.

**Aggregation Query:**

```
SELECT Agg-Op(Col) FROM T WHERE selection-condition
```

- The *Agg-Op* may be an aggregation operator such as SUM, COUNT, AVG.
- *T* is the table being queried
- *Col* may be any numeric measure column of *T*
- *selection-condition* decides which tuples should be involved in the aggregation.

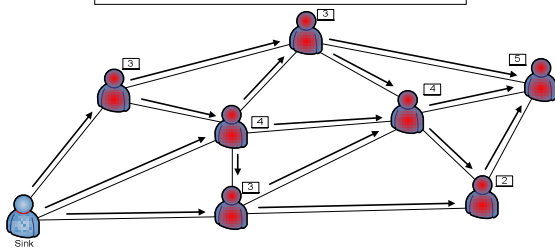
# Approximate Aggregation Queries

- Queries can be more efficiently answered using approximate aggregates where:

- Timeliness is more important than accuracy
- The exact solution cannot be computed in a reasonable amount of time or the picture may change
- It is not possible to reach of all of the peers for querying

# Exact Query Processing

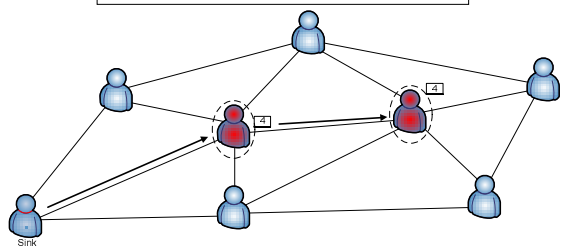
**Aggregation Query:** SELECT AVERAGE(Col) FROM T



Accuracy = 100%, Average = 3.43, 3 Hops, 7 Data Transmissions

# Approximate Query Processing

**Aggregation Query:** SELECT AVERAGE(Col) FROM T

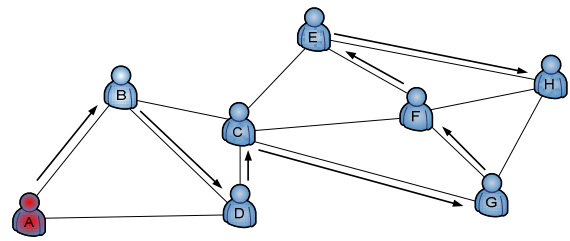


Accuracy = 83.3%, Average = 4, 2 Hops, 2 Data Transmissions

Why is it so difficult to compute aggregation queries in unstructured P2P topologies?

Lets look at an example...

## P2P Challenges



- No sense of data location (Which peers to search)
- Bandwidth constraints (Peers behind slow connections)
- Transient network (peers enter/leave network)

## Contributions

- Introduce the problem of approximate query processing in P2P databases
- Online sampling techniques for P2P databases
- Attempt to draw uniform random samples
- Two-phase sampling approach
- Extensive experimental results

## “On the fly” Sampling

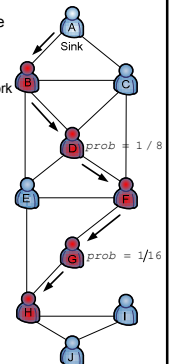
- **Online Samples:** samples drawn “on the fly” upon encountering a query used for a single query
- **Pre-computed Statistics:** samples such as random samples and histograms are computed by scanning the database and used for several queries
  - Cannot scan entire P2P repository
  - No centralized storage exists (No location for pre-computed values to reside)
  - Pre-computed values quickly become stale in dynamic topologies
- Problems with uniform random samples
  - Peers with higher degree are skewed towards giving higher probability
  - Peers may contain correlated data giving unrepresentative samples

## Approximate Query Cost

- We want to approximate the queries in such a way that latency and bandwidth are minimized
- We approximate latency as the number of tuples sent over the network for a given query
- Cost of sending samples back to the sink is only one hop (the location of the sink can be pushed to each peer)
- Overhead of visiting a peer dominates the cost of computing aggregates

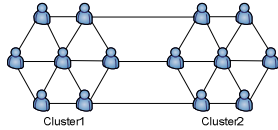
## Random Walks

- Seeking a random sample of the P2P database, we have to overcome the sub-problem of how to collect a random sample from the peers themselves
  - No peers knows the locations of all other peers in the network
  - Only aware of immediate neighbors
- Our solution is to use: Markov chain random walks [Gkantsidis et al., 2004]
  - Random walk starting at the sink
  - Each peer does not have an equal probability of selection
  - If walk is long enough, it will converge to a stationary distribution
  - The probability of any peer of an undirected graph given by the stationary distribution:
 
$$prob(p) = \frac{degree(p)}{2|E|}$$
  - Speed of convergence can be estimated using the second eigenvalue of the graph



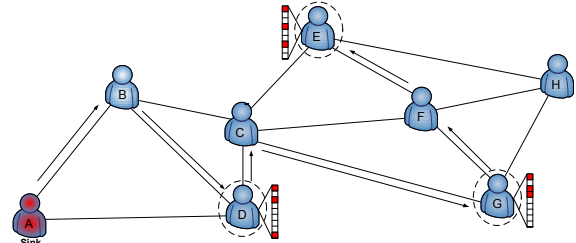
## Data Samples

- We introduce the user defined variables as follows:
  - $r$  (initial sample size): the initial number of tuples to sample from the network. Determines how many tuples to sample for the first phase
  - $t$  (tuples per peer): the number of tuples to sample per peer. Size of the parameter determines how many peers are sampled and how many tuples are sampled from each peer
  - $j$  (jump size): the number of peers to skipped between taking random samples



The Cut represents the edges connecting two clusters  
The smaller the cut size, the more difficult it becomes for a random walk to reach both clusters

## Example



- $r$  (initial sample size) = 9
- $t$  (# tuples sampled from each peer) = 3
- $j$  (number of peers to skip between selections) = 1

## A Technique for Sampling

### Phase 1:

- Initiate a fixed-length random walk from the sink
- Retrieve information from the visited peers
  - Number of tuples in the peer
  - Estimate Aggregate of selected tuples
- Send information back to the sink
- Analyzed skewed nature of the data that is distributed across the network
- Estimate how much more samples are required to meet required accuracy

### Phase 2:

- Reinitiate random walk obtaining estimate remaining samples
- Retrieve and send information back the sink (same as phase 1 steps 2 & 3)
- Compute final answer

## Sampling Theorems

- Consider a fixed-size sample of peers  $S = \{s_1, s_2, \dots, s_m\}$  where each  $s_i$  is from  $P = \{p_1, p_2, \dots, p_n\}$  the set of peers
- $\Delta_{req}$  is the desired error threshold
- Let the aggregate (COUNT) for a peer  $s$  be  $y(s) = \sum_{u \in S} y(u)$  where  $y(u) = 1$  if  $u$  satisfies the selection condition, otherwise 0
- Let  $y$  be the exact answer for the query  $y = \sum_{p \in P} y(p)$
- The query comes with a desired error threshold  $\Delta_{req}$  then  $|y' - y| \leq \Delta_{req}$  where  $y'$  is the estimated aggregate
- Consider the quantity  $y'$  defined as follows:
  - $y'$  is the weighted estimate from all the selected peers  $y' = \frac{\sum_{s \in S} y(s)}{m}$
- Theorem (Unbiased Estimator Theorem):**  $E[y'] = y$ ,  $y'$  is an unbiased estimator
- Theorem (Std Error Theorem):**  $Var[y'] = \frac{\sum_{p \in P} \left( \frac{y(p)}{prob(p)} - y \right)^2 prob(p)}{m} = \frac{C}{m}$ 
  - Notice dividing the variance by  $N^2$  provides the square error of  $y'$

## Cross-Validation

- Randomly Partition  $S$  into two samples  $S_1$  &  $S_2$

$$y_1' = \frac{\sum_{s \in S_1} y(s) / prob(s)}{m/2} \quad y_2' = \frac{\sum_{s \in S_2} y(s) / prob(s)}{m/2}$$

- Cross-Validation Error:  $CVError = |y_1' - y_2'|$

**Theorem (Cross-Validation Theorem):**  $E[CVError^2] = 2E[(y'' - y)^2]$

- Estimating sample size:

$$Var[y'] = C/m \rightarrow m' = C/Var[y']$$

$$m' = \frac{C}{Var[y']} = \frac{C}{\Delta_{req}^2} = \frac{2E[(y'' - y)^2]}{\Delta_{req}^2} = (m/2) \times \left( \frac{CVError^2}{\Delta_{req}^2} \right)$$

## Putting it All Together

- Each phase contains both computation and sampling components

### Phase 1:

- Random select peers  $S = \{s_1, s_2, \dots, s_m\}$
- Compute  $y(s_i)$  at each selected peer sending the result and number of tuples in the peer back to the sink
- Compute the cross-validation error for  $y_1'$  and  $y_2''$
- estimate number of tuples  $m'$  to meet the required accuracy

### Phase 2:

- Random sample  $m' - m$  tuples
- Compute  $y(s_i)$  at each selected peer sending the result and number of tuples in the peer back to the sink
- Compute final aggregate  $y' = \frac{\sum_{s \in S'} y(s) / prob(s)}{m'}$  estimate at the sink

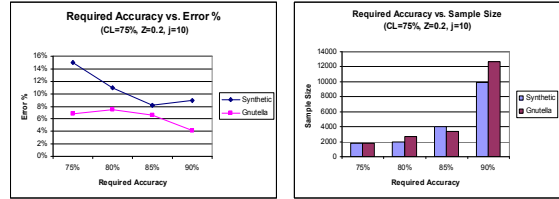
## Experimental Setup

- Implementation
  - Java 5.0 utilizing the Jung library (<http://jung.sourceforge.net>) for graph generation
  - Our implementation includes both synthetic and real-world Gnutella topologies
    - Synthetic network: 10,000 peers, 1,000,000 tuples
    - Gnutella network: 22,556 peers, 2,200,000 tuples
  - Experiments were ran on AMD dual Opteron 2.0 GHz processors with 2GB of RAM.
  - All of our experiments were generated from five independent experiments and averaged for each individual configuration
- Input Parameters
  - Required Accuracy** [ $\Delta_{req}$ ]: This parameter defines the maximum allowed error for the estimated answer.
  - Tuples Sampled per Peer** [ $t$ ]: This parameter defines the number of tuples to sample from each selected peer.
  - Jump Size** [ $j$ ]: This parameter defines the Number of peers to pass over before selecting the next peer for sampling.
  - Initial Sample Size** [ $r$ ]: This parameter defines the initial number of tuples to acquire from the
- Metrics For Evaluating our Approach
  - Cluster Level [CL]**: The clustering of data within the peers
  - Skew [Z]**: The skew determines the slant in frequency distribution of distinct values the data.

Benjamin Arai University of California, Riverside

## Accuracy Results

- For our experiments the algorithm meets the required accuracy for all experiments
- As the required accuracy increases, the required sample size increases as well
- The number of peers sampled is a fraction of the number of peers in the network

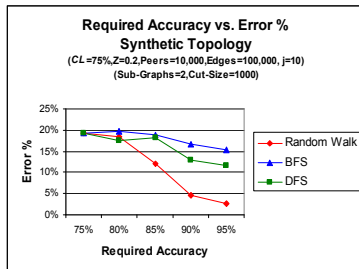


Error % and sample size as the required accuracy is increased for the COUNT technique

Benjamin Arai University of California, Riverside

## Samples Per Peer & Random Walks

- Our technique achieves lower error % than naïve approaches such as BFS and DFS

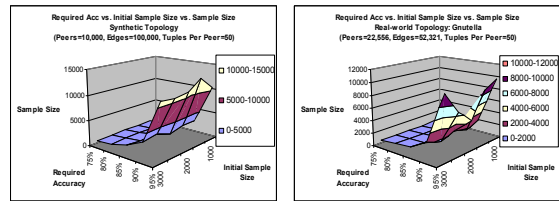


Comparison of various naïve and random walk approaches.

Benjamin Arai University of California, Riverside

## Required Accuracy

- In order to meet the required accuracy of the user, as the required accuracy increases the number of samples required increases



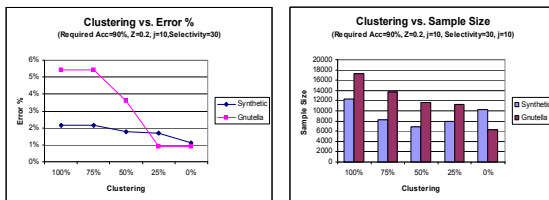
As the required accuracy increases, the estimated sample size increases for the COUNT technique

Benjamin Arai University of California, Riverside

## Effects of Clustering

Clustering determines how permuted an originally sorted dataset is distributed across the network

- No clustering (0%) implies data is not sorted partitioned into peers
- Complete clustering (100%) implies data is sorted partitioned into peers
- As the clustering decreases, the error % and the estimated sample size decreases



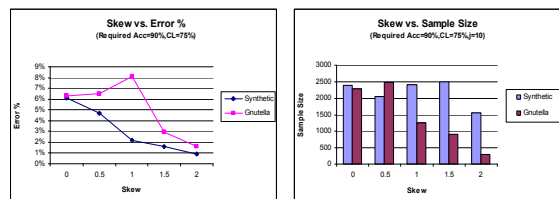
The error % and sample size as a function of the clustering for the COUNT technique

Benjamin Arai University of California, Riverside

## Effects of Skew

Skew determines the slant in the frequency distribution of distinct values.

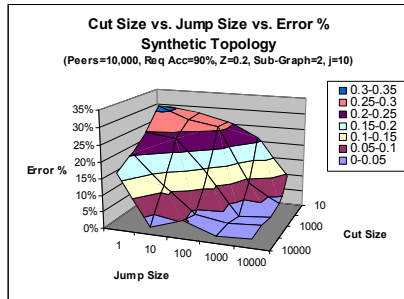
- Low skew gives even distributions (frequency of values)
- High skew give distorts the distribution of values
- As the skew increase, it becomes easier for the algorithm to meet the required accuracy of the user, requiring smaller sample sizes



As skew increases, the accuracy and estimated sample size decreases for the COUNT technique

Benjamin Arai University of California, Riverside

## Network Topology and Cut Size



Comparison of various levels of clustering with the jump size and error % for the SUM technique

Benjamin Arai University of California, Riverside

## Users Guide to Setting Parameters

- The below bullets offer guidelines for obtaining optimal results using our method, but are not required to obtain results.
  - Jump Size [j]:** We have found that as the cut size decreases between sub-graphs, the jump size  $j$  needs to be increased. Based upon the real-world topology data tested a value of 25 suffices for most cases.
  - Initial Sample [r]:** We have found that for most data layouts an initial sample size of 1000 tuples suffices for computing the estimated samples sizes for the ranges of required accuracy and selectivity tested.
  - Tuples Sampled per Peer [t]:** We have found the number of tuples sampled per peer obtains best results when between 10 to 20 tuples.

Benjamin Arai University of California, Riverside

## Approximating the Median

- Computing the *MEDIAN* in a distributed fashion
  - Select  $m$  peers at random using random walk.
  - Each peer  $s_j$  computes its median  $med_j$  and sends it to the sink, along with  $prob(s_j)$ .
  - The sink randomly partitions the  $m$  medians into two groups of  $m/2$  medians, *Group1* and *Group2*.
  - Let  $med_{g1}$  be the *weighted median* of Group1, i.e., such that the following is minimized
 
$$abs \left( \sum_{\substack{med_j \in Group1 \\ med_j < med_{g1}}} 1/prob(s_j) - \sum_{\substack{med_j \in Group1 \\ med_j > med_{g1}}} 1/prob(s_j) \right)$$
  - Find the error between the median of Group2 (say  $med_{g2}$ ) and the weighted rank of  $med_{g1}$  in Group2. i.e., let

$$C = abs \left( \sum_{\substack{med_j \in Group2 \\ med_j < med_{g1}}} 1/prob(s_j) - \sum_{\substack{med_j \in Group2 \\ med_j > med_{g1}}} 1/prob(s_j) \right) / (m/2)$$

- Select additional  $C/N_{peers}$  peers using random walk and return the weighted median of the medians of the additional peers.

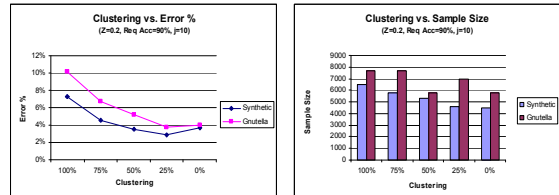
Benjamin Arai University of California, Riverside

## Median Approximation Results

Clustering determines how permuted an originally sorted dataset is distributed across the network

- No clustering (0%) implies data is not sorted partitioned into peers Complete
- clustering (100%) implies data is sorted partitioned into peers

As the clustering decreases, the error % and the estimated sample size decreases



Effects of clustering on the error percentage and sample size for the MEDIAN technique

Benjamin Arai University of California, Riverside

## Related Work

- Approximate query processing in centralized environments [Babcock et al., 2003, Chaudhuri et al., 2001, Acharya et al., 1999, Li et al., 2003]
  - Block-level sampling [Chaudhuri et al., 2004]
  - Histogram estimation [Chaudhuri et al., 1998]
- Computed Approximate Aggregates in P2P database
  - Gossip Protocol [Boyd et al., 2004, Boyd et al., 2005, Kempe et al., 2003]
  - Mercury [Bharambe et al., 2004]
- Markov Sampling [Gkantsidis et al., 2004, King et al., 2004, Bharambe et al., 2004]
- P2P Models
  - Structured overlay networks [Ratnasamy, et al., 2001, Rowstron et al., 2001, Stoica et al., 2001]
  - Unstructured overly networks [Chu et al., 2000, Milojicic et al., 2002, Zeinalipour-Yazti et al., 2005]

Benjamin Arai University of California, Riverside

## Conclusions & Future Work

### Conclusions

- Adaptive sampling-based techniques for the approximate answering of ad-hoc aggregation queries in P2P databases
  - Minimal number of messages sent over the network
  - Provides tunable parameters to maximize performance for various network topologies
- Varying few parameters our algorithm successfully computes aggregates within a given required accuracy.
- We present extensive experimental evaluations to demonstrate the feasibility of our solutions for both synthetic and real-world topologies.

### Future Work

- Hybrid solutions involving both pre-computation and "on the fly" samples
- Decision support and data analysis in P2P databases
- Hybrid sampling techniques building upon the Gossip protocol

Benjamin Arai University of California, Riverside

## Approximating Aggregation Queries in Peer-to-Peer Networks

Benjamin Arai – ([barai@cs.ucr.edu](mailto:barai@cs.ucr.edu))  
Gautam Das – ([gdas@cse.uta.edu](mailto:gdas@cse.uta.edu))  
Dimitrios Gunopulos – ([dg@cs.ucr.edu](mailto:dg@cs.ucr.edu))  
Vana Kalogeraki – ([vana@cs.ucr.edu](mailto:vana@cs.ucr.edu))

<http://www.cs.ucr.edu/~barai>