Scalable Spatial Crowdsourcing: A study of distributed algorithms

Abdullah Alfarrarjeh [#], Tobias Emrich [*], Cyrus Shahabi [#]

[#] Integrated Media Systems Center, University of Southern California

[*] Dept. of Computer Science, Ludwig Maximilian University of Munich

What is Spatial Crowdsourcing?

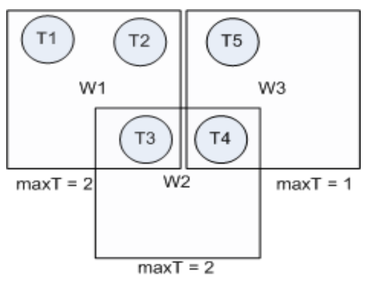
•The concept of crowdsourcing refers to outsourcing tasks to a set of people, known as the workers.

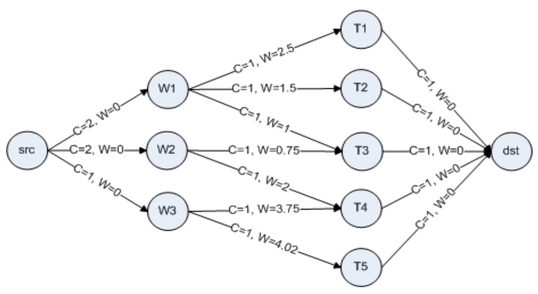
•In Spatial crowdsourcing, tasks are specified by a location so workers are required to be physically present at a task location in order to perform the corresponding task.

- Example: take a picture of a particular building

•Given a set of spatial tasks and workers, the spatial crowdsourcing server performs the task assignment. The spatial task assignment can be formulated as a minimum-cost maximum flow problem where the cost is the travel distance.

- Time Complexity: *O((m + n log n) n C); n: the number of vertices, m: number of edges, C: maximum edge capacity*





Why Scalable Spatial Crowdsourcing?

•There is a need for an online system where a large number of workers and tasks arrives in a short amount of time.

•The requester of tasks targets to have all of the tasks performed in a short amount of time.

•Current studies focused on the maximization of assigned tasks rather than the runtime of the assignment in a large-scale settings.

Existing Scalable Spatial Crowdsourcing:

•The task assignment can be executed in a MapReduce infrastructure. This approach provides an exact solution but the gained speedup is not high because of the following drawbacks:

- It requires collecting spatial tasks, workers and finding the candidacy relations between them (i.e., constructing the graph) then transferring the constructed graph to the MR distributed file system to be partitioned across servers.
- It poses a high volume of intercommunication when partitioning/re-arranging data to generate the final result.
- The input must be transferred to DFS and the output should retrieved from DFS

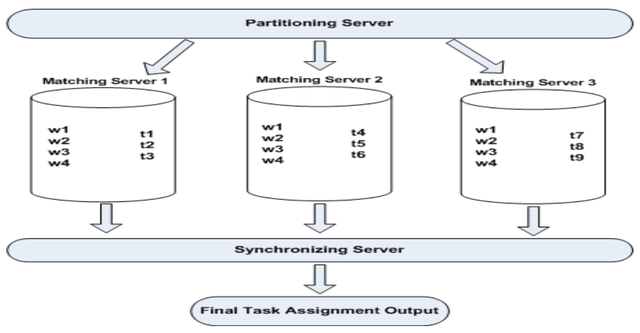
Our Scalable Spatial Crowdsourcing Solutions:

- We propose a class of approaches that utilizes an online partitioning method to reduce the problem space across a set of servers and construct the bipartite graph independently and solve the assignment in parallel.
- Our approaches solve the spatial task assignment approximately but competitive to the exact solution

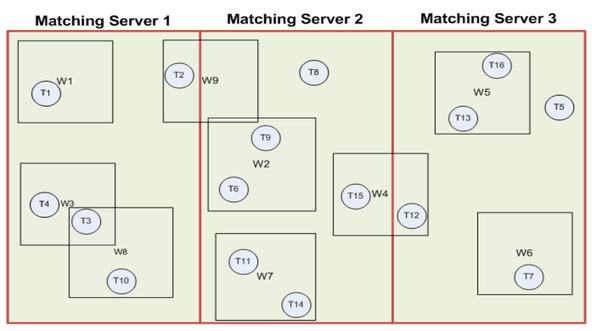
Scalable Spatial Crowdsourcing Approaches

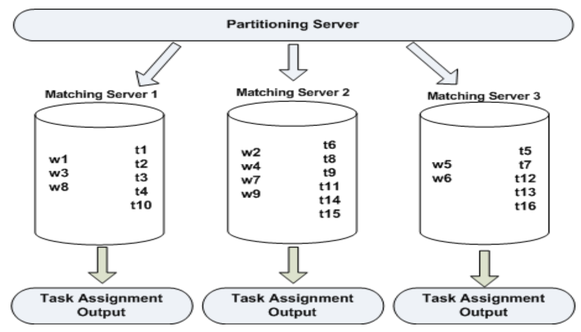
•Random Task Partitioning with Worker Replication Approach (*RTP-WR-A*):

- Incoming Workers are replicated to all M-Servers
- Incoming tasks are partitioned across M-Servers in round-robin fashion



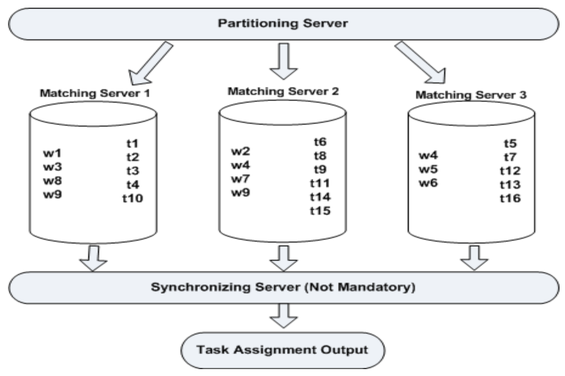
- **Spatial Partitioning Approach (SP-A):**
  - Each M-Server is assigned to a geographical area.
  - Both tasks and workers are spatially partitioned based on their location





•Spatial Partitioning with Worker Replication Approach (SP-WR-A):

- Incoming Workers are partitioned based on their work regions. If the worker region overlaps with the geographical area of multiple M-Servers, the worker is replicated to all of them.
- Incoming tasks are partitioned based on their location



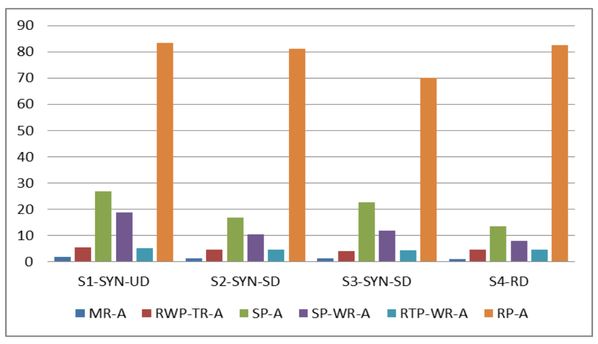
•Other Variations:

•*Random Partitioning Approach (RP-A)*: It is a variant of SP-A. The tasks and workers are partitioned randomly.

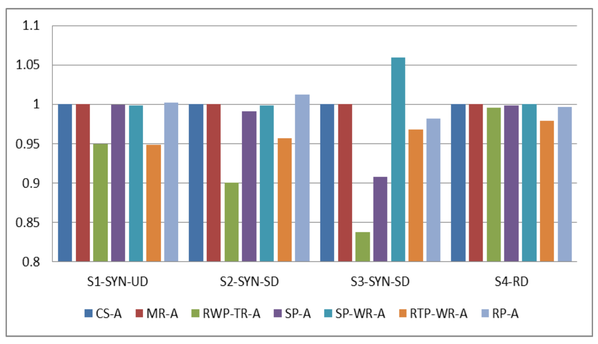
•Random Worker Partitioning with Task Replication Approach (RWP-TR-A): It is a variant RTP-WR-A. The tasks are replicated and workers are partitioned.

Experiments and Evaluation

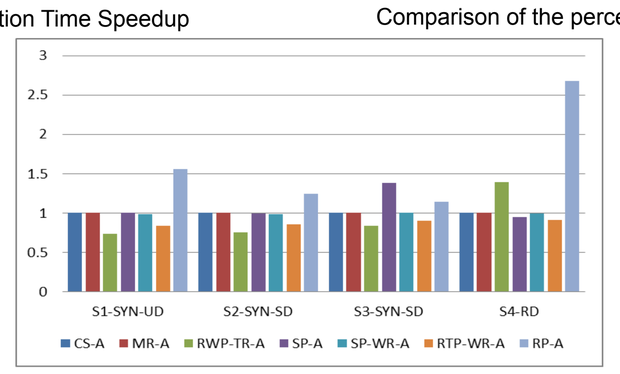
- We simulated our experiments in four settings:
  - S1-SYN-UD: tasks and workers are uniformly distributed.
  - S2-SYN-SD: workers are distributed uniformly, tasks are distributed normally. S3-SYN-SD is similar to S2-SYN-SD but the Gaussian centers of workers dataset were chosen close to the geographical borders of M-Server areas.
  - S4-RD: workers and tasks are obtained from Brightkite and Gowalla datasets.
- Evaluation metrics:
  - The execution time speed-up
  - The percentage of assigned tasks
  - The percentage of assignment cost per assignment pair.



Comparison of the Execution Time Speedup



Comparison of the percentage of assigned tasks



Comparison of the assignment cost per assignment pair

Conclusion

- A single centralized system is not capable to cope with a rising number of workers and tasks arriving in short amount of time.
- We thus evaluated several approaches that differ in the distribution and handling of the incoming data.