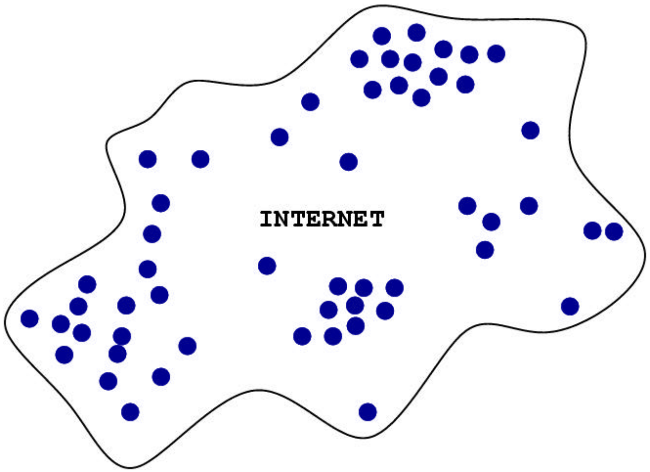
Latency-Driven Replica Placement

Michal Szymaniak   Guillaume Pierre   Maarten van Steen

Vrije Universiteit Amsterdam
The Netherlands
{michal,gpierre,steen}@cs.vu.nl

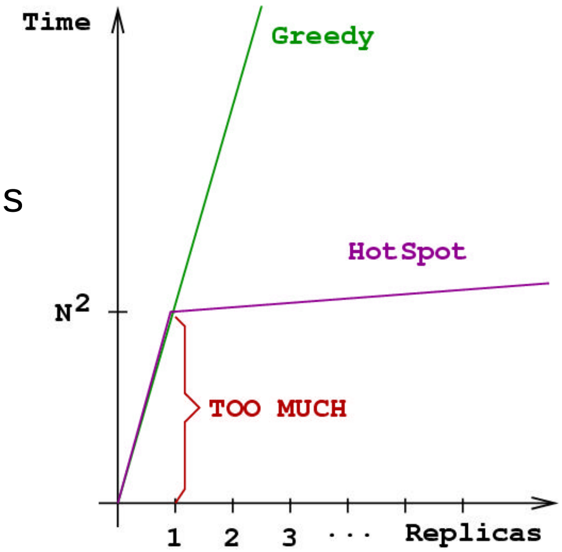
Problem Description



- Large distributed system
  - Thousands+ of nodes
- Wide-area network
  - Internet
- Node = client + server
  - Nodes can host content
- Most popular content is replicated
  - Thousands of possible replica locations
- Where to place replicas efficiently?
- Efficient = minimal average client-to-replica latency
- Clients always use their closest replicas

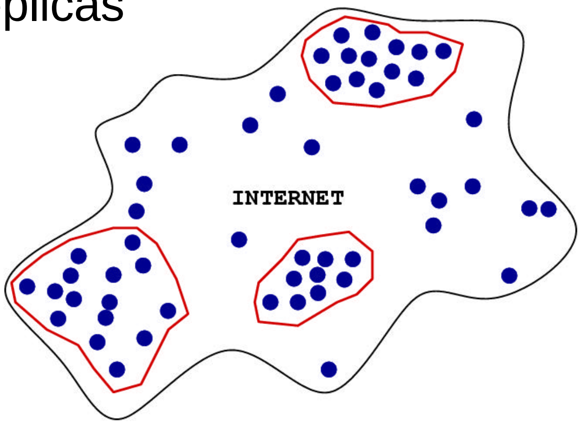
Current Solutions

- Greedy
  - Place replicas one-by-one
  - Each time evaluate all possible locations
  - Good placement quality
  - O(K*N^2), K replicas, N candidate locations
- Hot-Spot
  - Compute load generated by each location
  - Place replicas in K most active locations
  - Slightly worse quality than Greedy
  - O(N^2+min(N*logN,K*N))
- Note:
  - O(N^2) is too much for large-scale systems
  - O(N^2) caused by all-pair latency calculations; can we get rid of them?



Our Two-Step Solution

- 1: Cluster locations; choose clusters for replicas
  - Clustered nodes close in terms of latency
- 2: Select nodes inside clusters
  - Current work



- Identify clusters efficiently (HotZone)
  - Model latencies such that clustering is cheap
  - We use Global Network Positioning (GNP)

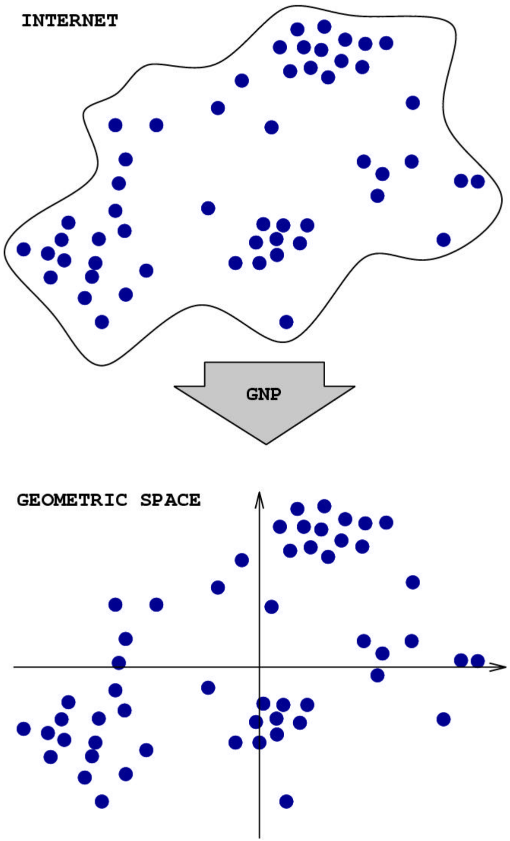
- HotZone identifies clusters in O(N*max(logN,K))

Agenda

- Efficient Latency Modeling
  - Global Network Positioning
- Cluster Identification
- Performance
  - Placement Quality
  - Computation Times
- Conclusion

Efficient Latency Modeling

- Global Network Positioning (GNP):
  - Internet == M-dimensional geometric space
  - Nodes == M-dimensional positions
  - Latencies == distances between positions
- GNP can be run efficiently even in large-scale systems
  - Previous work
- So: we play with points in geometric space
- How to identify clusters of points?



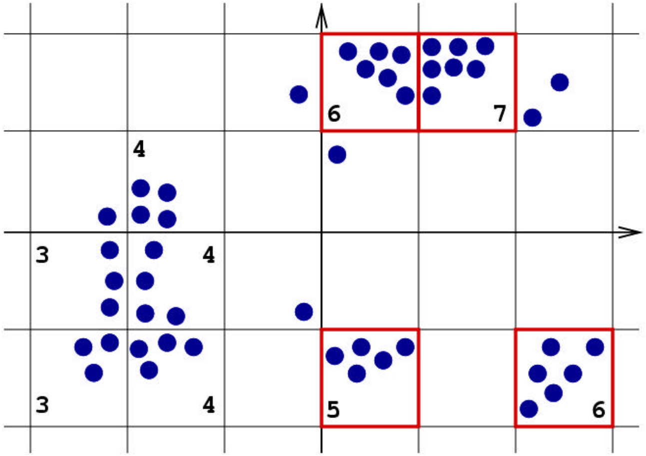
Cluster Identification

- Divide space into M-dimensional hypercubes (cells)
- Cell density = number of nodes inside cell

- We are done!
  Take most dense cells as clusters!

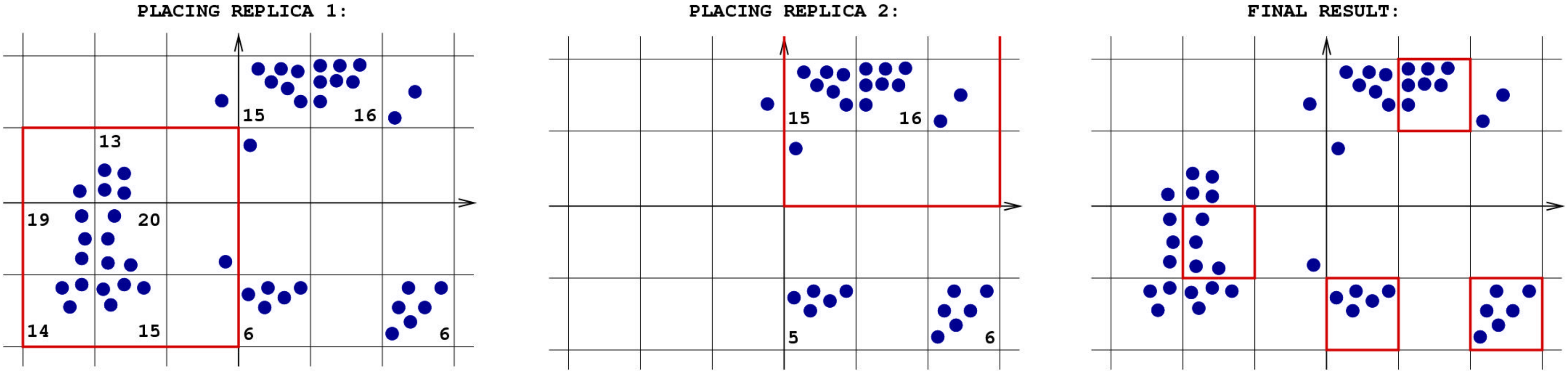
- Not quite:
  - We could cut clusters into pieces..
  - ..which can be too small..
  - ..to be assigned replicas :-(

- What can we do about it?



Fixing Split Clusters

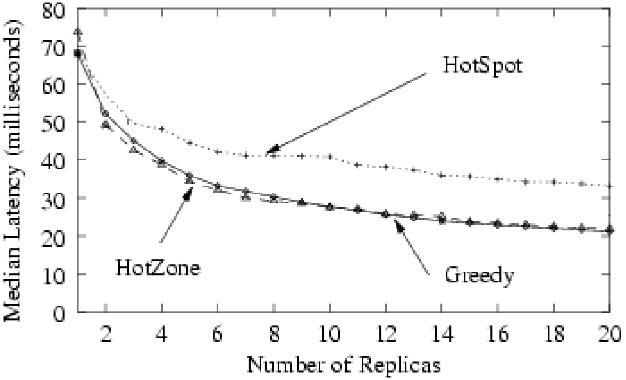
- Solution: adjust density definition
  - Cell density = the number of nodes INSIDE + AROUND the cell.
  - After placing each replica - remove nodes that replica shall service!



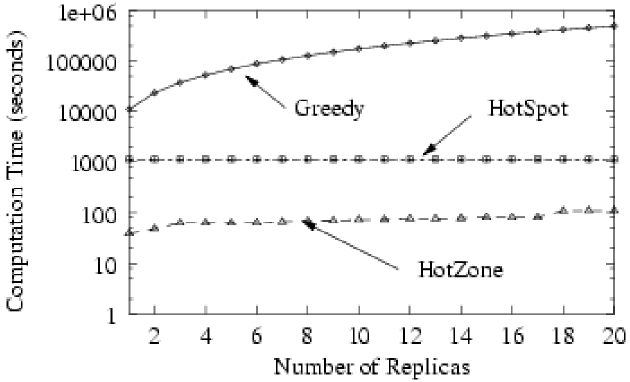
- What if we cannot unambiguously identify dense cells?
  - Wrong cell size; adjust it to node distribution.

Performance

- Placement Quality..



..and Computation Time



- Tested for 64k nodes (clients == possible replica locations)

Conclusions and Future Work

- Two-step replica placement for large-scale systems:
  - 1. Cluster locations according to latency; choose biggest clusters
  - 2. Inspect chosen clusters to select nodes that will hold replicas
- First step - HotZone:
  - Relies on geometric system model provided by GNP
  - Identifies biggest node clusters at low cost: O(N*max(logN,K))
  - Preserves ultimate placement quality
- Second step - Current work:
  - Not so many nodes -- consider their individual properties
  - Clusters = virtual servers; they will dynamically manage local replicas

Thank you!

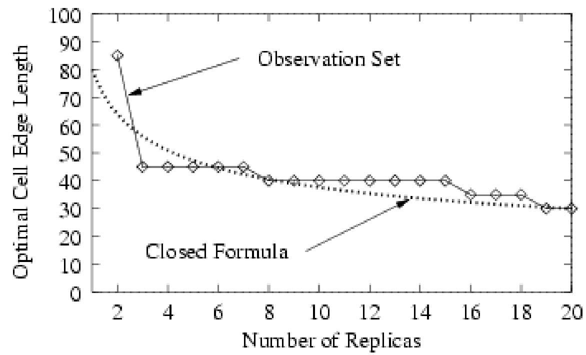
Questions?

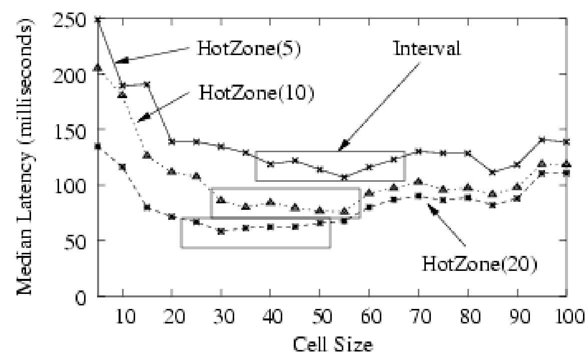
Extras: Complexity

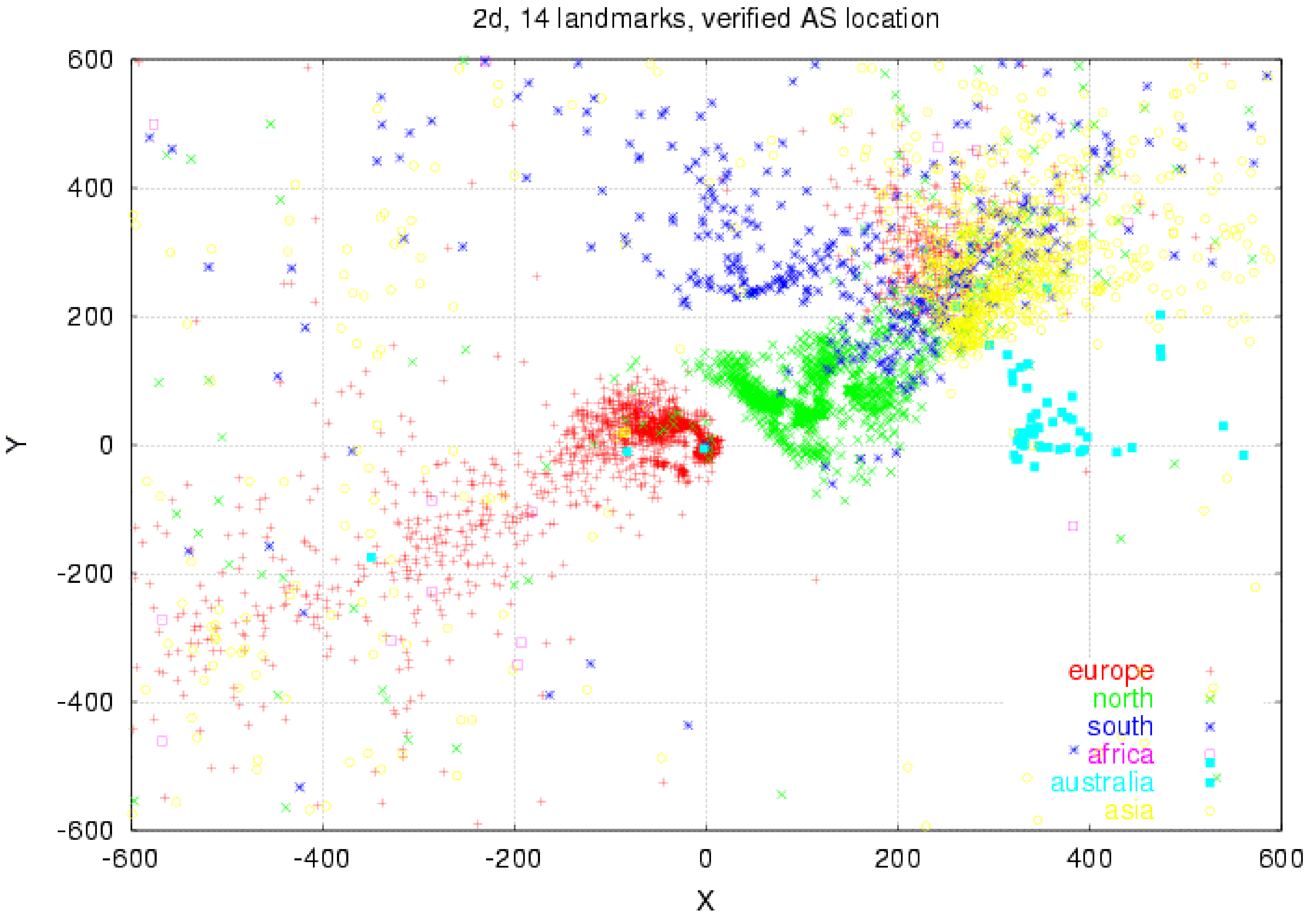
- Entry: we know positions of all N nodes
- Divide geometric space into O(N) cells: O(N)
  - For each position: O(1) to identify target cell
  - Cells identified by their center positions
- Calculate densities O(NlogN)
  - O(N) to calculate all cell densities
  - O(N) merges with neighbor densities
  - But: neighbor lookup costs O(logN) in our data structures
- Choose K clusters for replicas O(KN)
  - For each replica: O(N) to find most dense cell..
  - ..and O(logN) to remove that cell and its neighbors
- Total: O(N*max(logN,K))

Extras: Cell Size

- Cell size C intuitively depends on two factors:
  - node distribution (e.g., average inter-node distance D)
  - number of replicas to place K
- Let C=A*D/K^B; (A,B) - parameters
- Obtain (A,B) using non-linear regression:
  - Try all (C,D,K) combinations on a sample
  - Identify best C values for all (D,K) pairs
  - Assign (A,B) such that best C~= A*D/K^B
- Experiments:
  - A~1/8, B~1/3 for our sample
  - (A,B) will vary for other datsets
  - Still: placement quality resilient to small changes in A and B






2d, 14 landmarks, verified AS location
600
400
200
0
-200
-400
-600
Y
-600
-400
-200
0
200
400
600
X
europe
north
south
africa
australia
asia