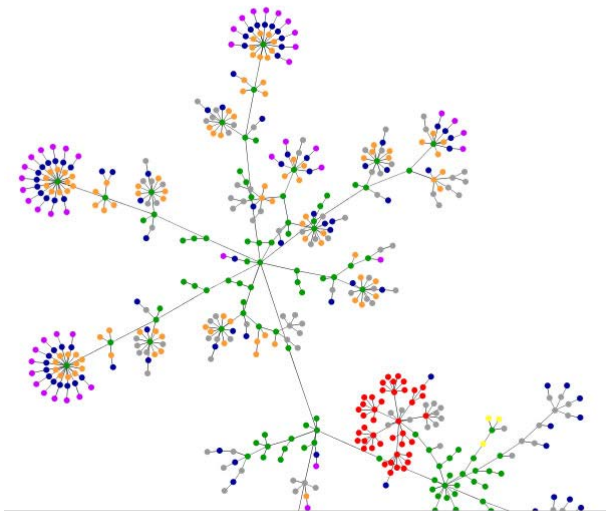
Interactive Graph Computing and Weibo Data

Wentao Han

Department of Computer Science and Technology,
Tsinghua University

Graph Data

- An important form of unstructured data
- And usually BIG
- Can represent rich data and relationships
  - Traffic networks
  - Web link graphs
  - Social networks
  - Protein interaction graphs



What Does Researchers Do When Doing Graph Analysis?

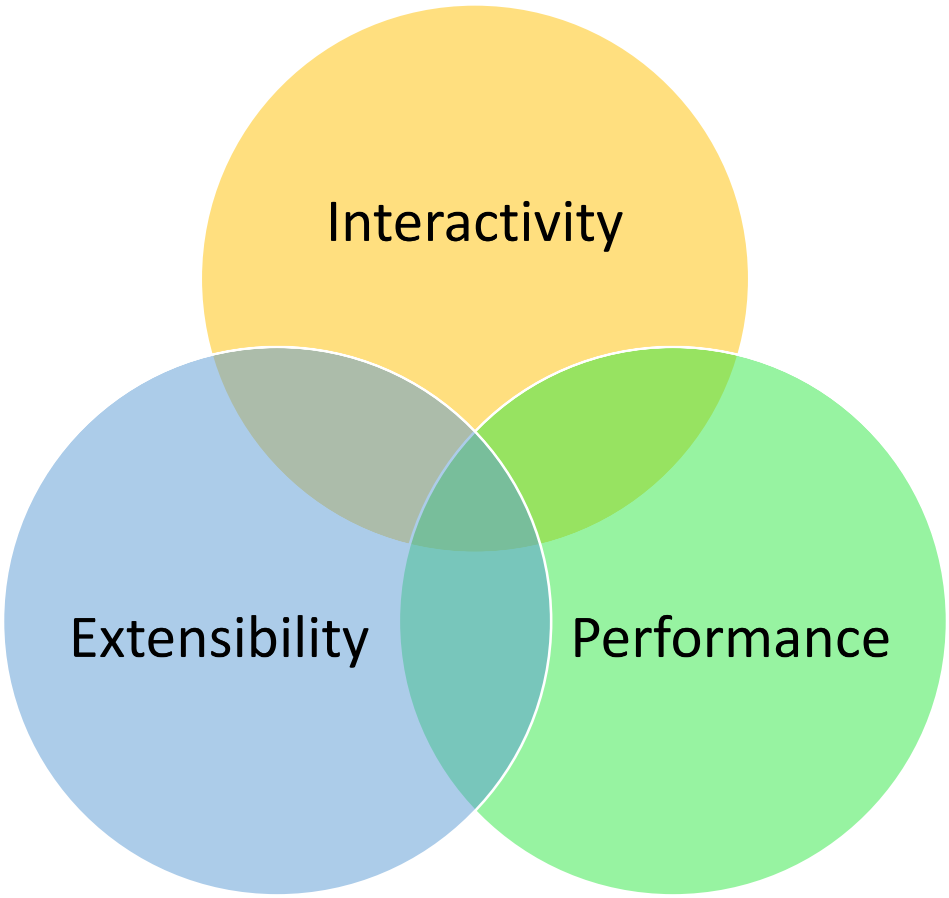
From the perspective of operations

- ETL (extract, transform, load)
- Run workflows composed of algorithms
- Improve by modifying algorithms or adjusting parameters

Pain Points in Graph Analysis

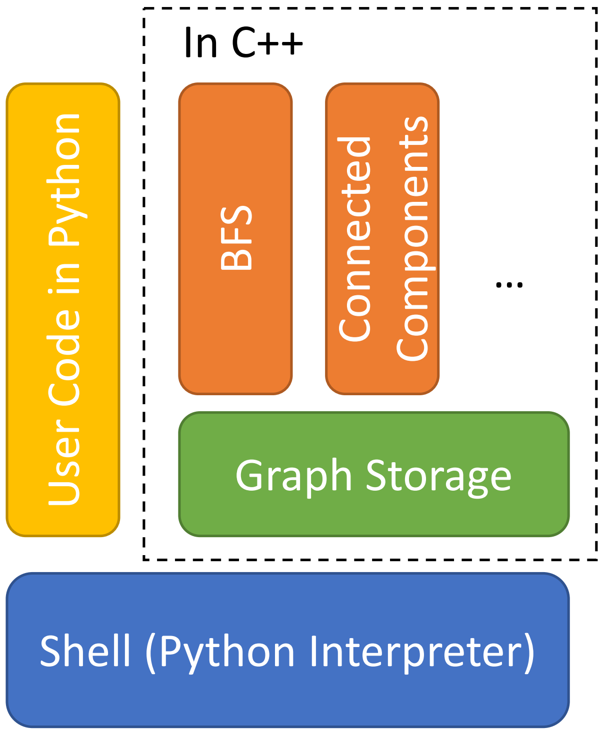
- Data is big: deploying a distributed processing system is costly (hardware, maintenance, power)
- Algorithms (or parameters) are changing: not only the actual operations, but also loading operations will take a lot of time (I/O bandwidth bottleneck)
- Workflows are changing, too: results from previous steps may affect what to do next
- Others: visualization, etc.

Requirements for Graph Analysis



Our Solution: Interactive Graph Computing

- Interactivity: provided by underlying Python shell
- Performance: implementing core data structures and algorithms in C++, in in-memory or out-of-core fashion
- Extensibility: provided by Python interpreter, extending in C++ (core code) or Python (glue code)



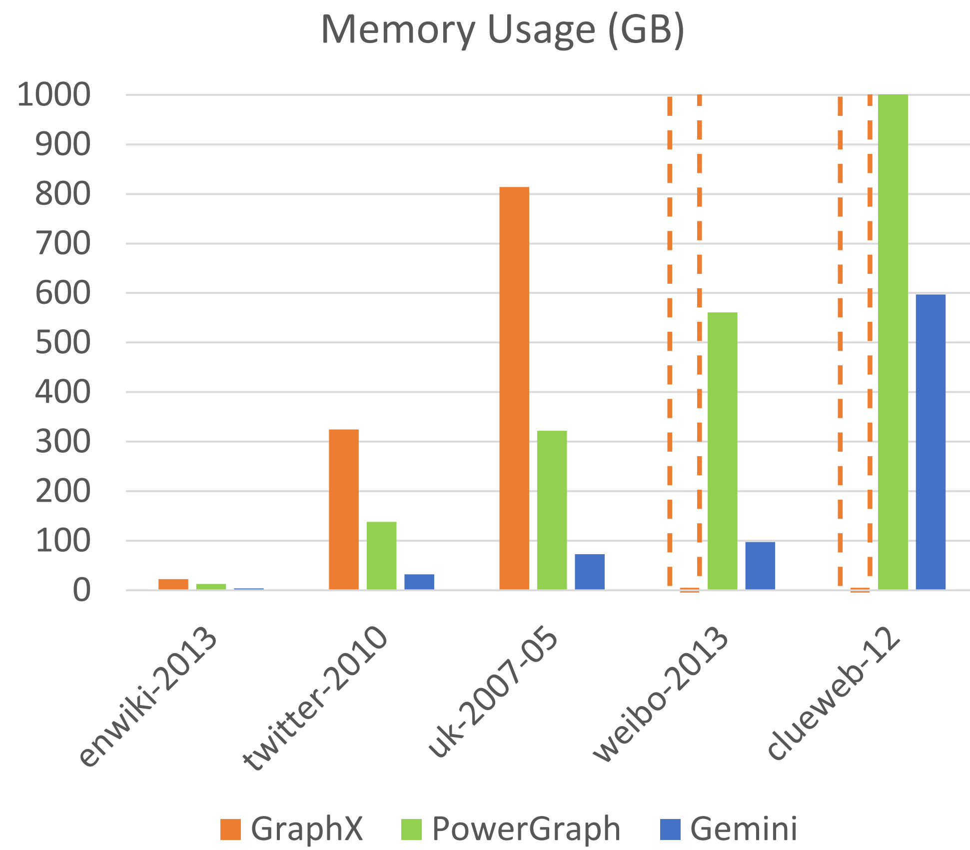
Architecture of Interactive Graph Computing Framework

Single Machine vs. Distributed Systems

Speed
100x less efficient

scalable system	cores	twitter	uk-2007-05
GraphChi [12]	2	3160s	6972s
Stratosphere [8]	16	2250s	-
X-Stream [21]	16	1488s	-
Spark [10]	128	857s	1759s
Giraph [10]	128	596s	1235s
GraphLab [10]	128	249s	833s
GraphX [10]	128	419s	462s
Single thread (SSD)	1	300s	651s
Single thread (RAM)	1	275s	-

Memory
20x less efficient

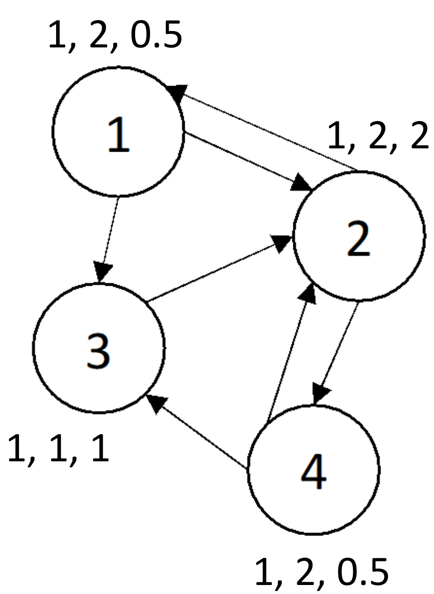


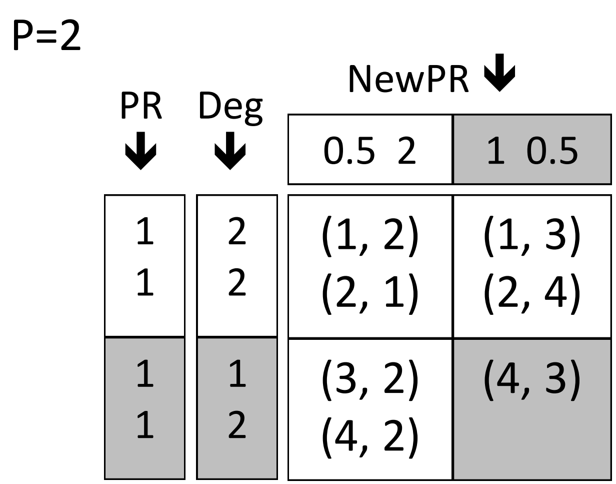
In-memory: Compact Structures

- Not using complex containers like trees or hashmaps
- Represent graph data in plain arrays
- For example, CSR (compressed sparse row) format, space complexity O(V + E)
- Choose data width economically, and even use bit compression (trade-off between time and space)

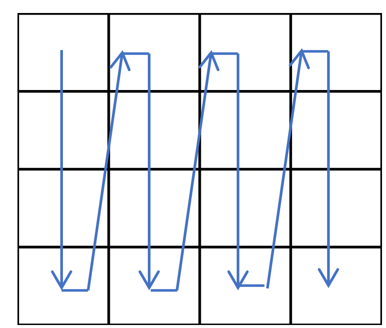
Out-of-core: Layout and Scheduling

- Layout: to increase data locality
- Scheduling: to reduce I/O amount





Stream Block (2, 2)

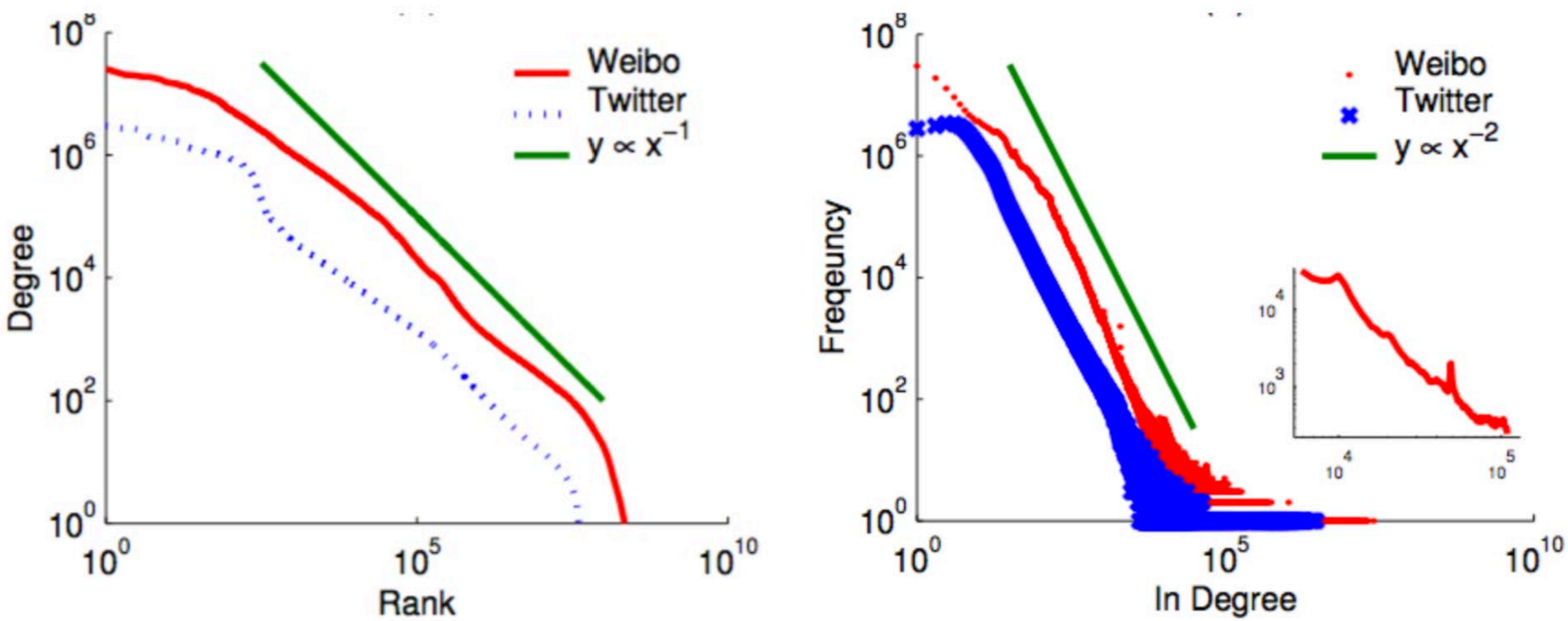


Case Study: Weibo Data (Compared to Twitter)

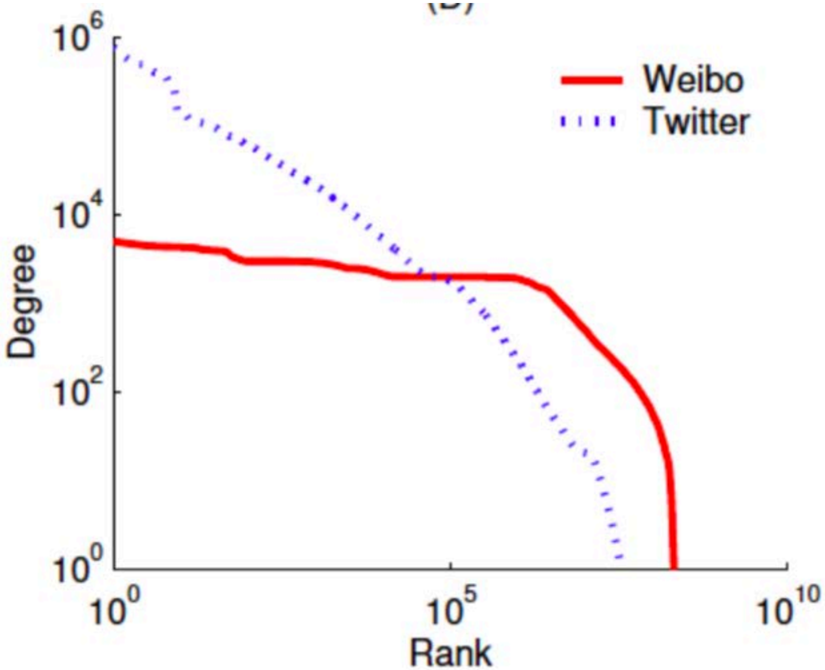
Weibo Data Profile

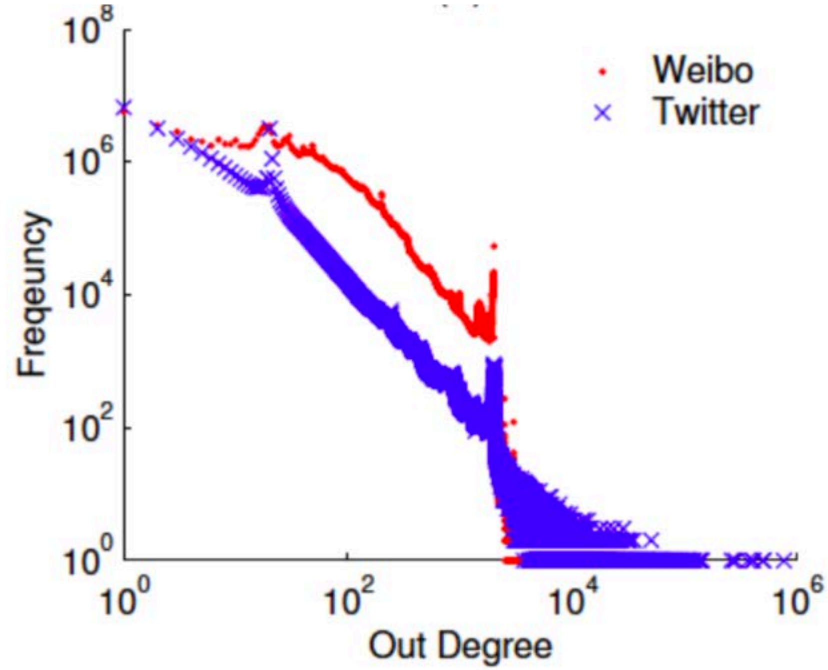
- We crawled Weibo data from 2012 to 2013
  - VERY DIRTY WORK!
  - Don’t want to and can’t do it again
- The data contains
  - User profiles (222 million)
  - Following relationships (27 billion)
  - Weibo posts (about 10TB in MongoDB’s data format)
  - Largest OSN available for research at that time
- Compared to Twitter data in 2009

In-degree Distribution

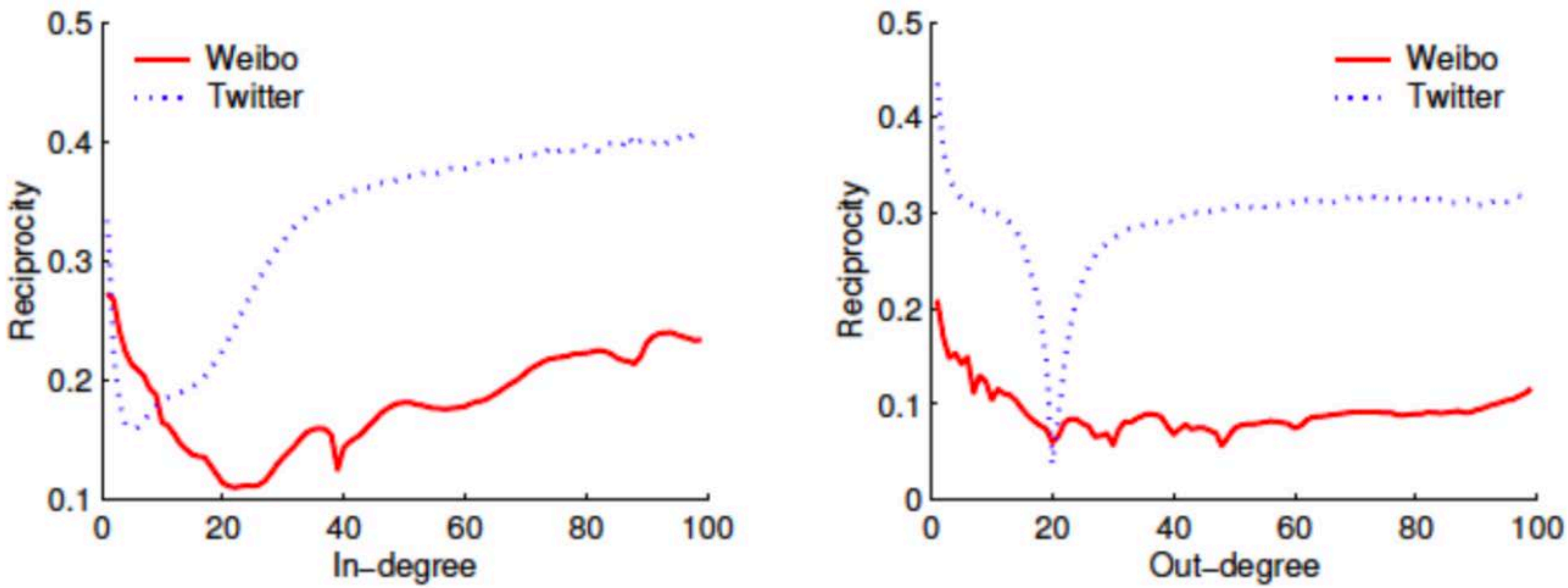


Out-degree Distribution

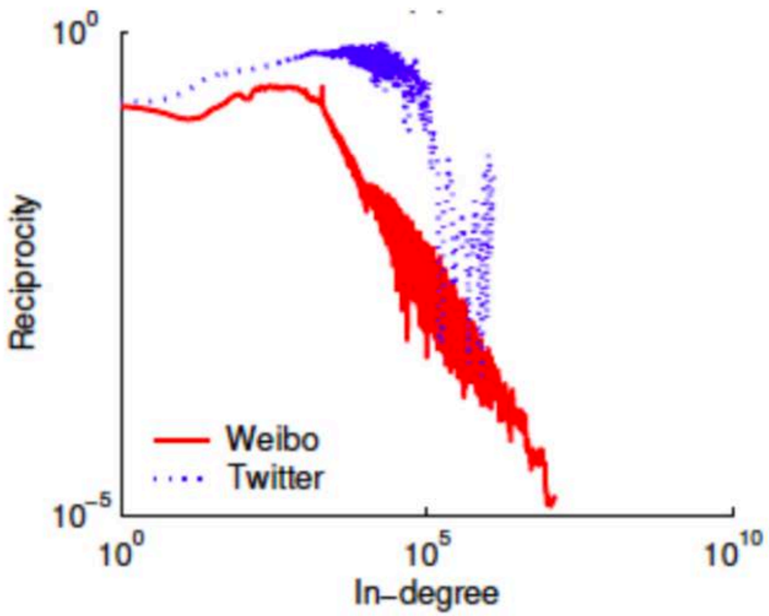


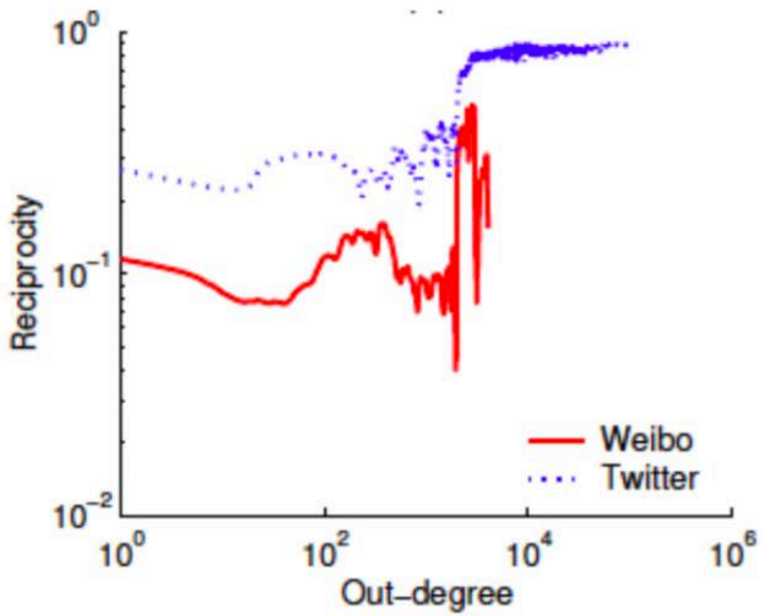


Reciprocity (degree less than 100)

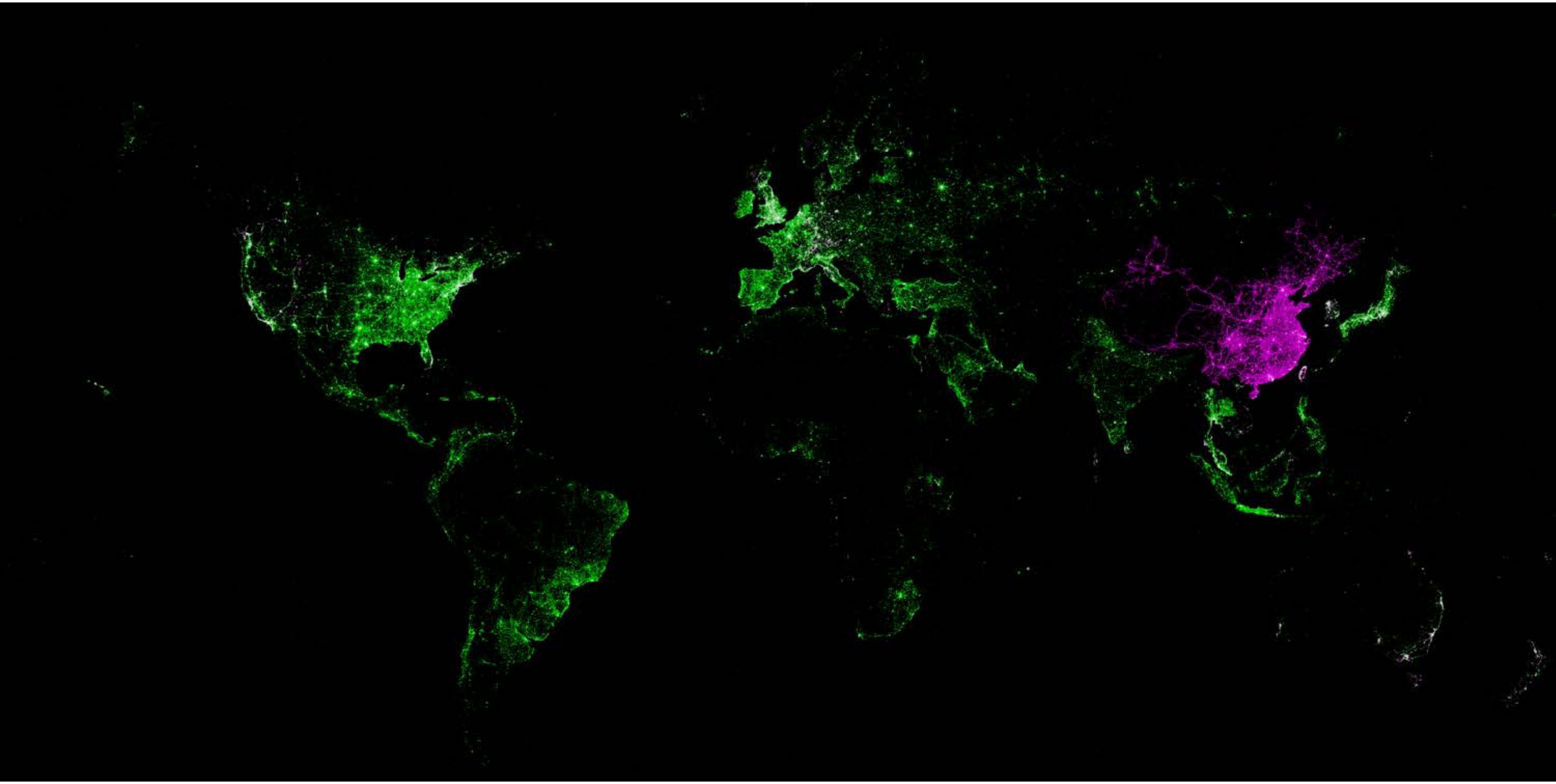


Reciprocity (overall)





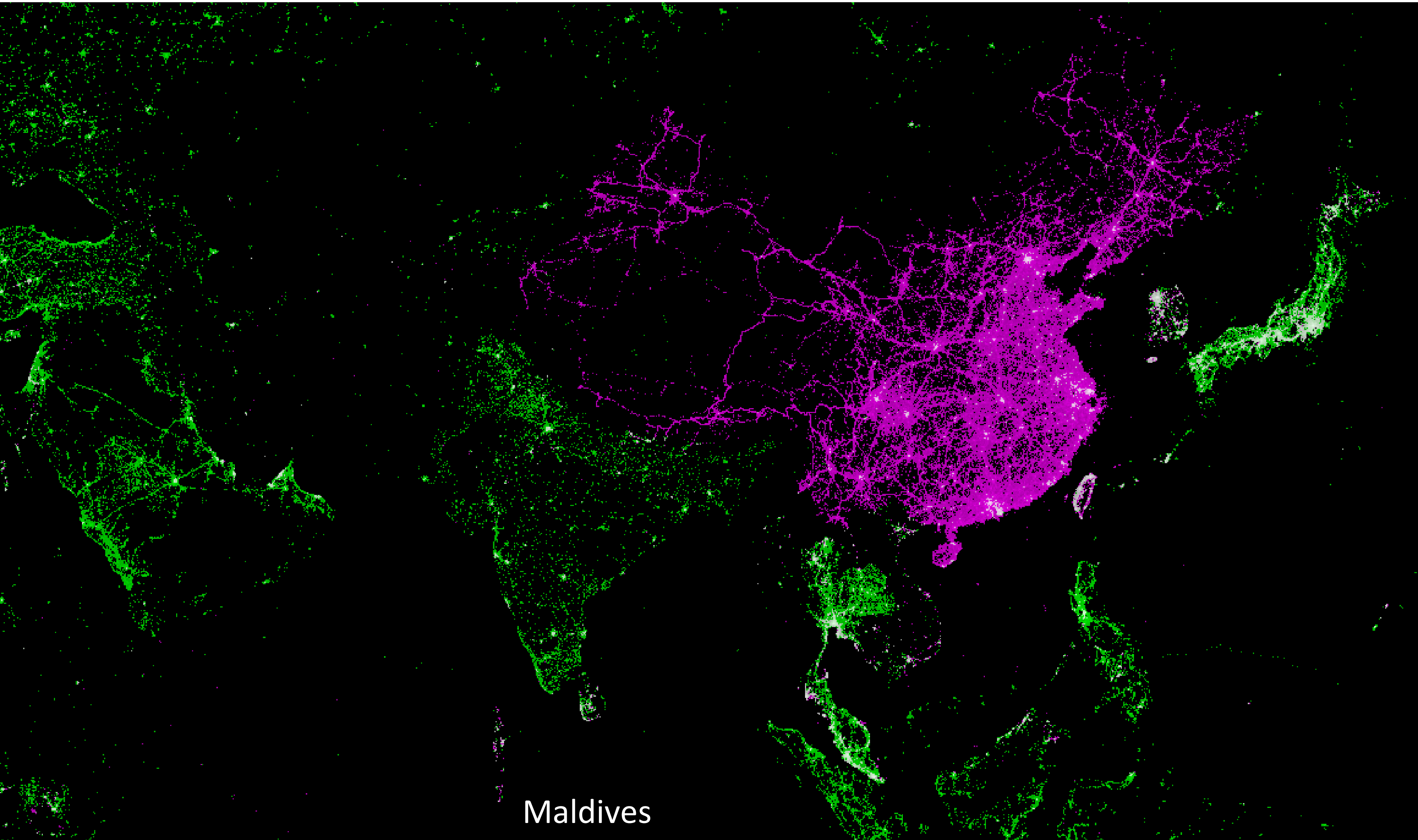
Posting Heatmap



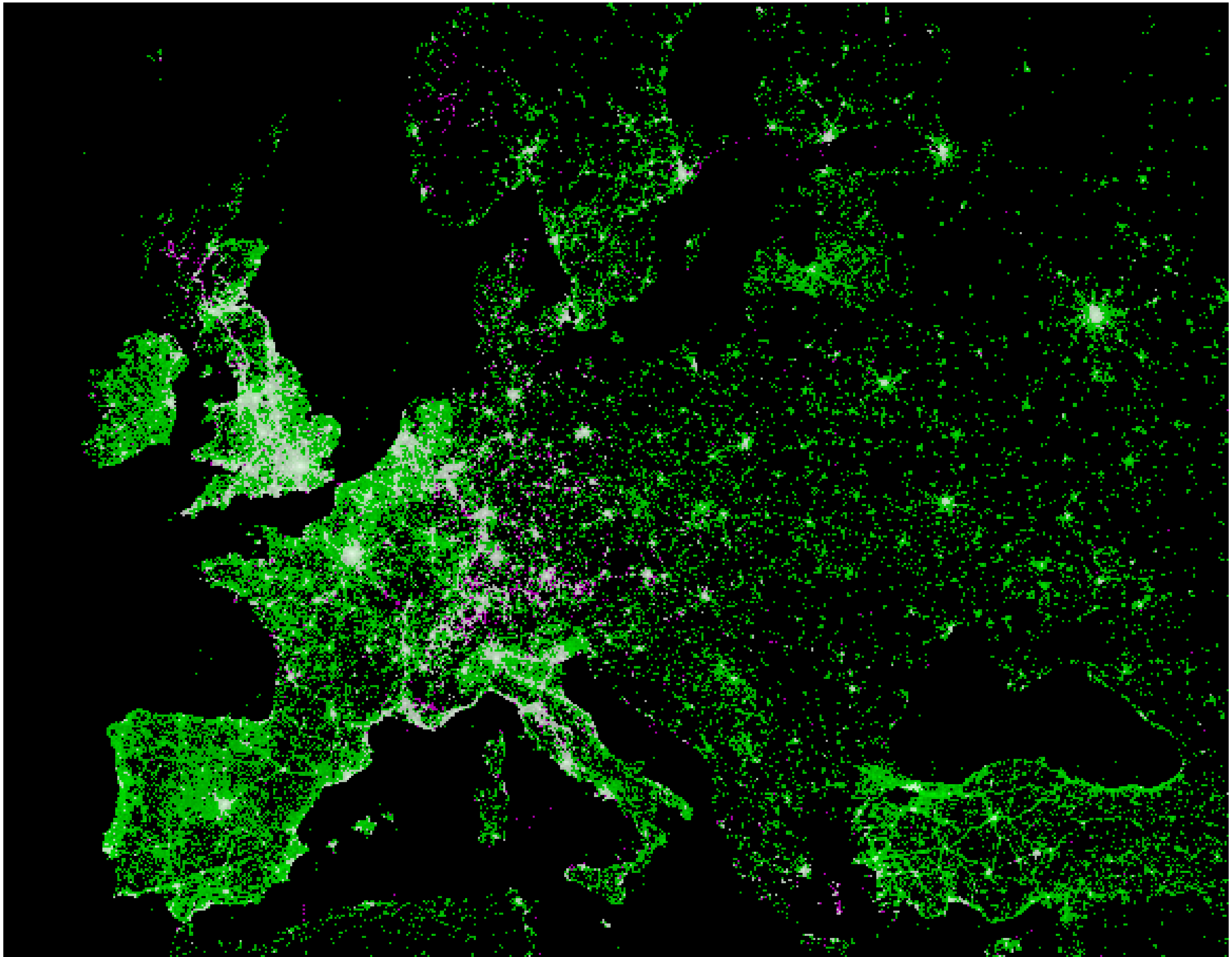
Heatmap of Weibo (2012) and Twitter (2009) over the world

Weibo (37 million posts) in magenta, and Twitter (42 million posts) in green

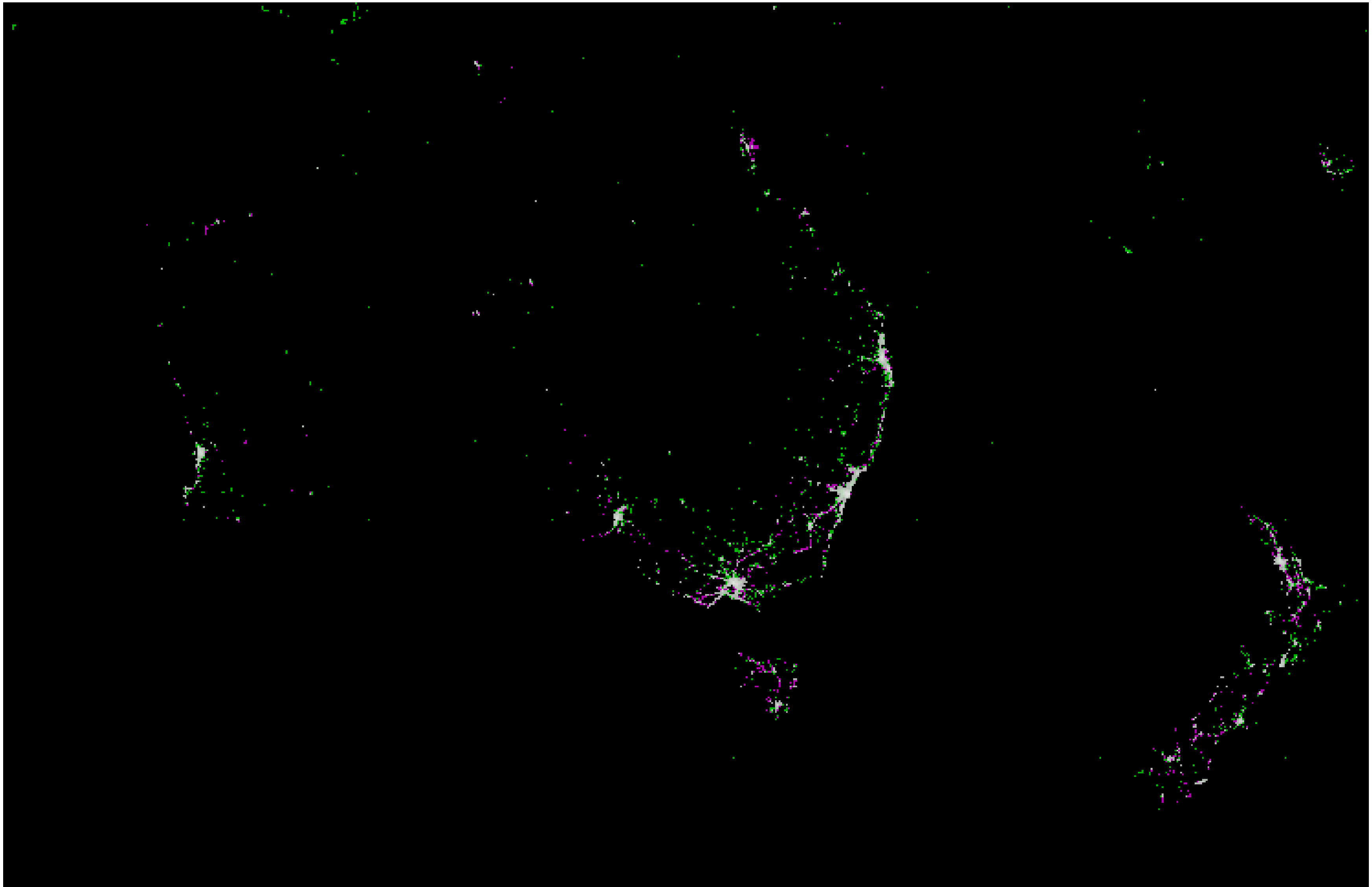
Heatmap of Asia



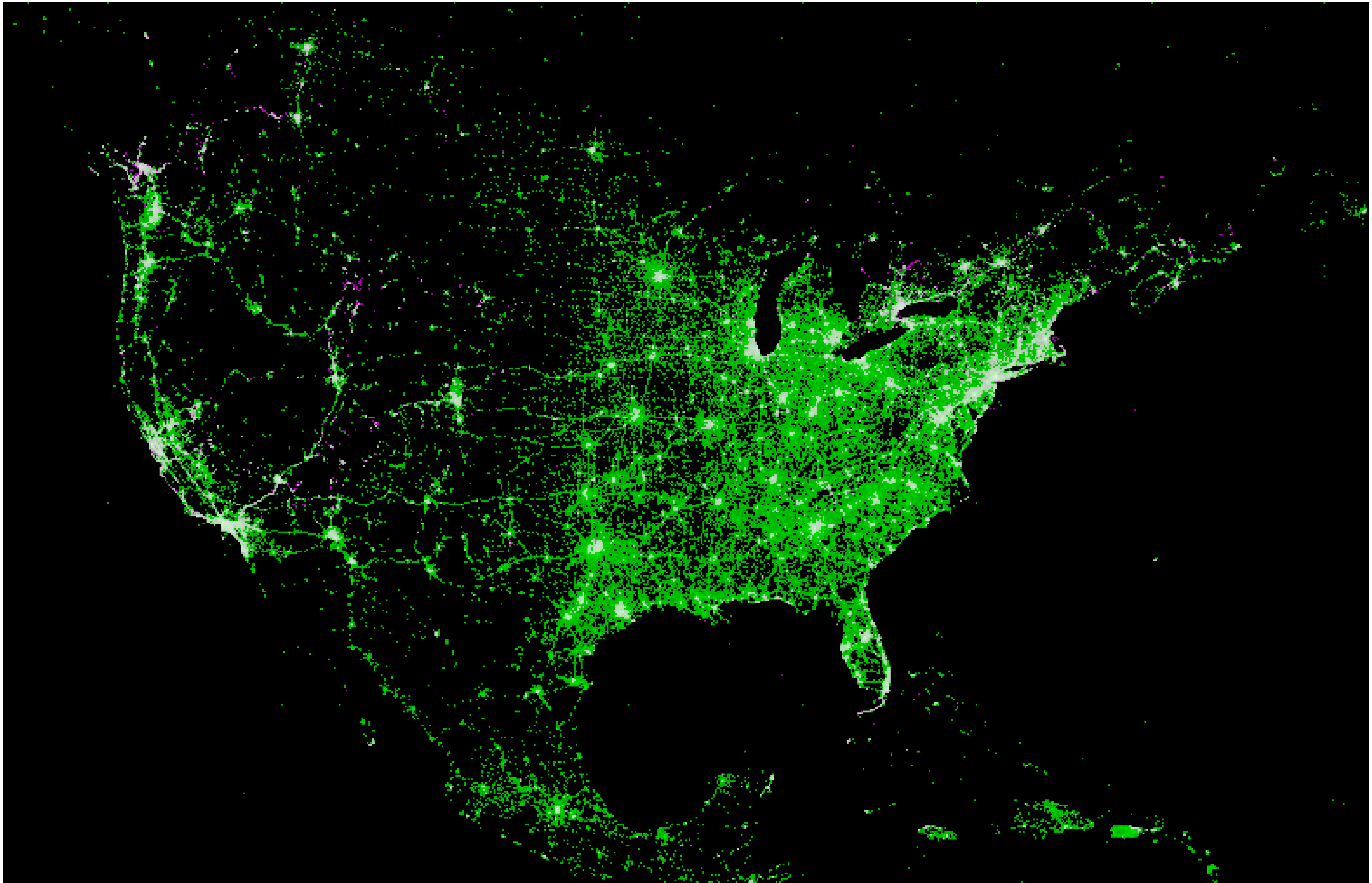
Heatmap of Europe



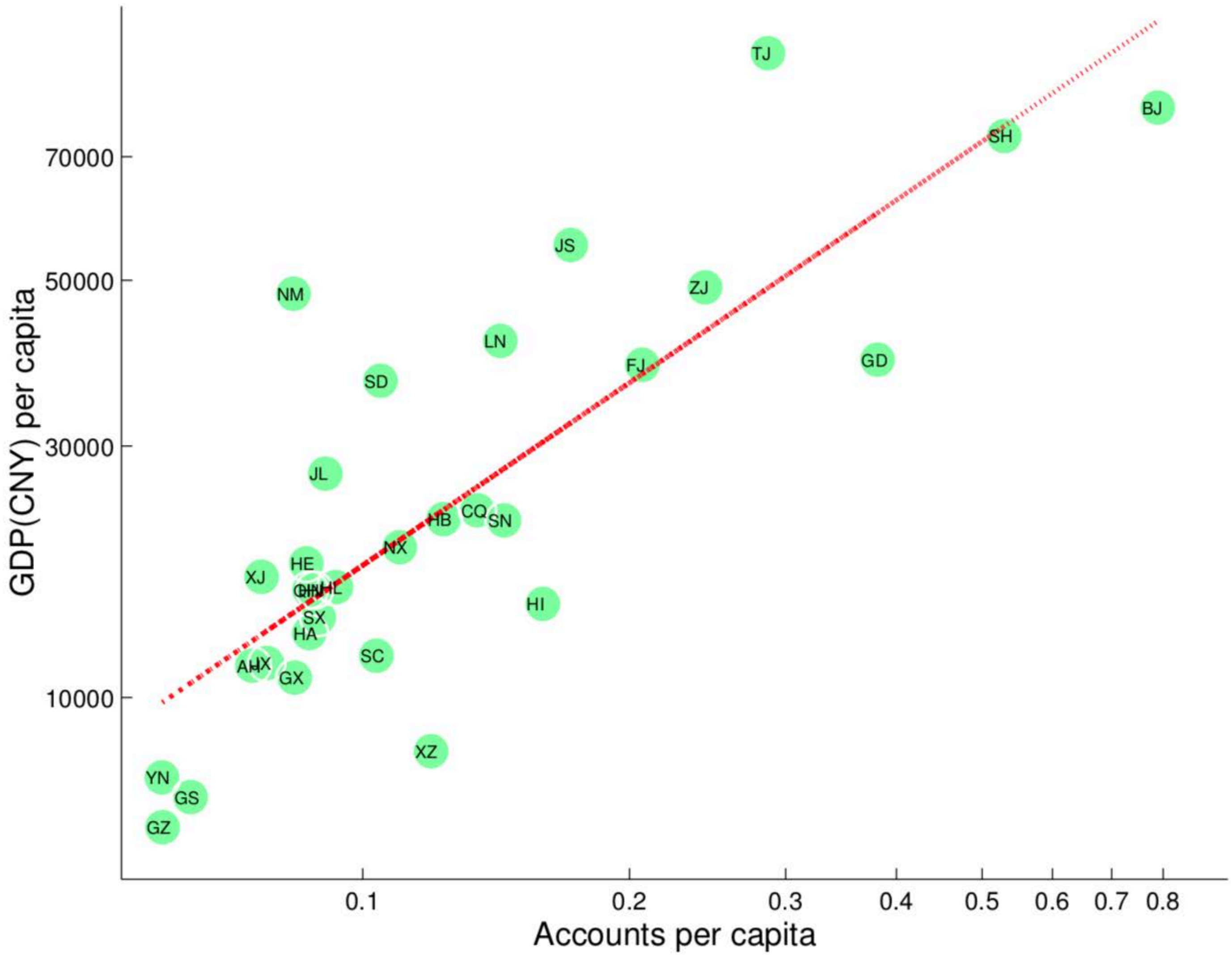
Heatmap of Oceania



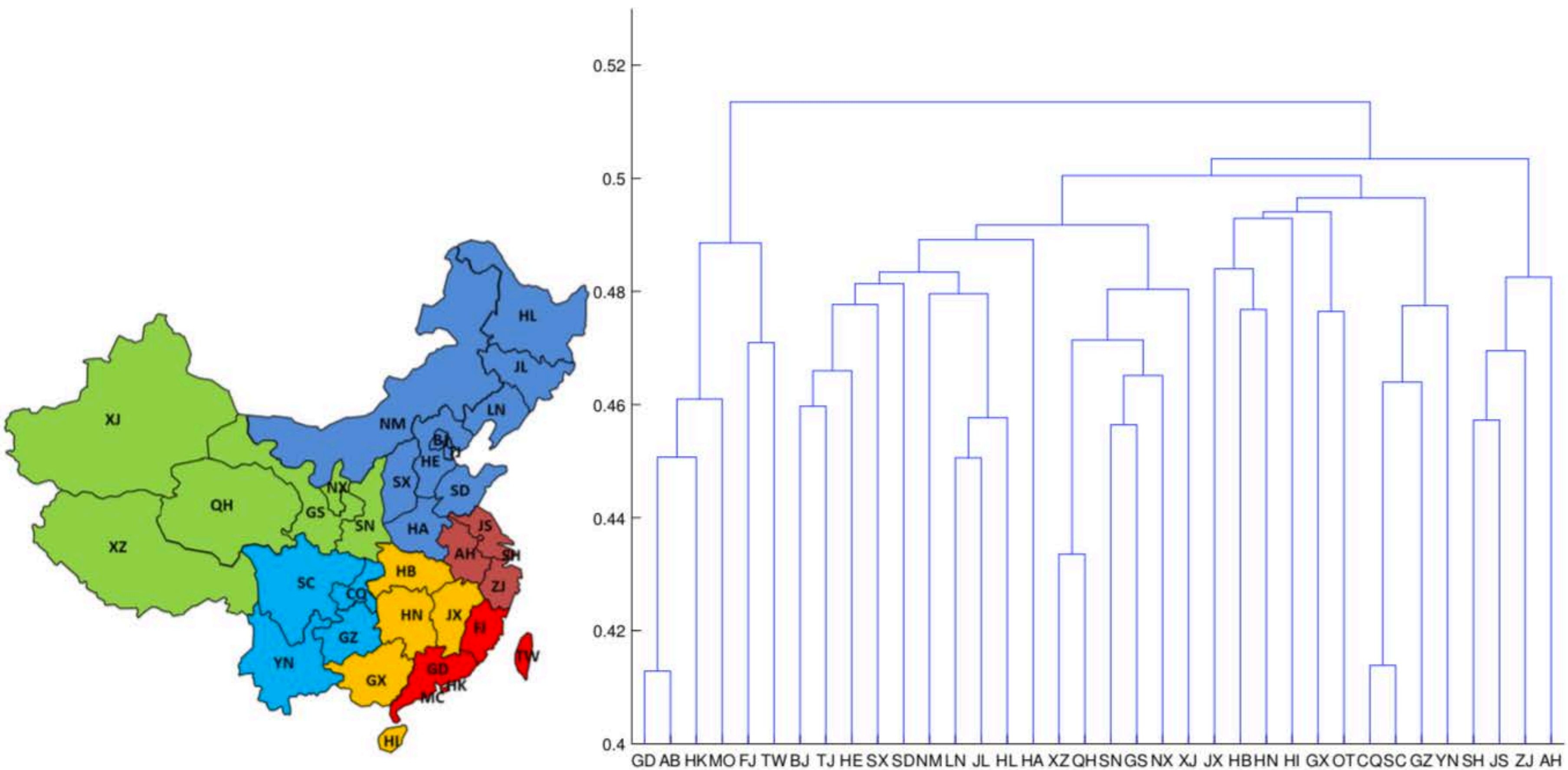
Heatmap of North America



Weibo Adoption Rate



Connections between Regions



result of running hierarchical agglomerative clustering

Take-away Messages

- Interactive graph computing is in demand.
- Single-machine systems could be great; don't use distributed systems unless have to.
- Some interesting results from Weibo with Twitter