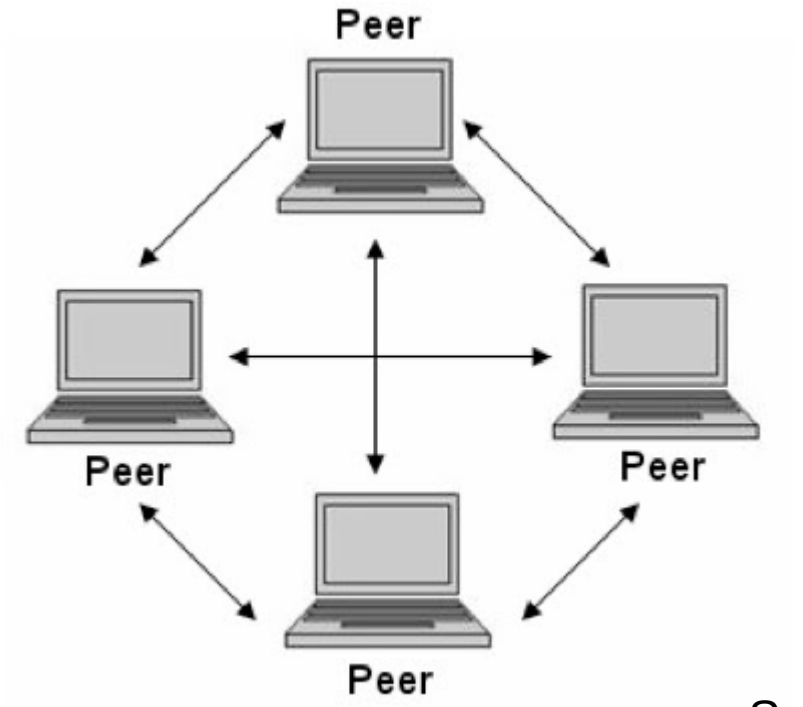
CS698F Advanced Data Management

Instructor: Medha Atre

Distributed Storage

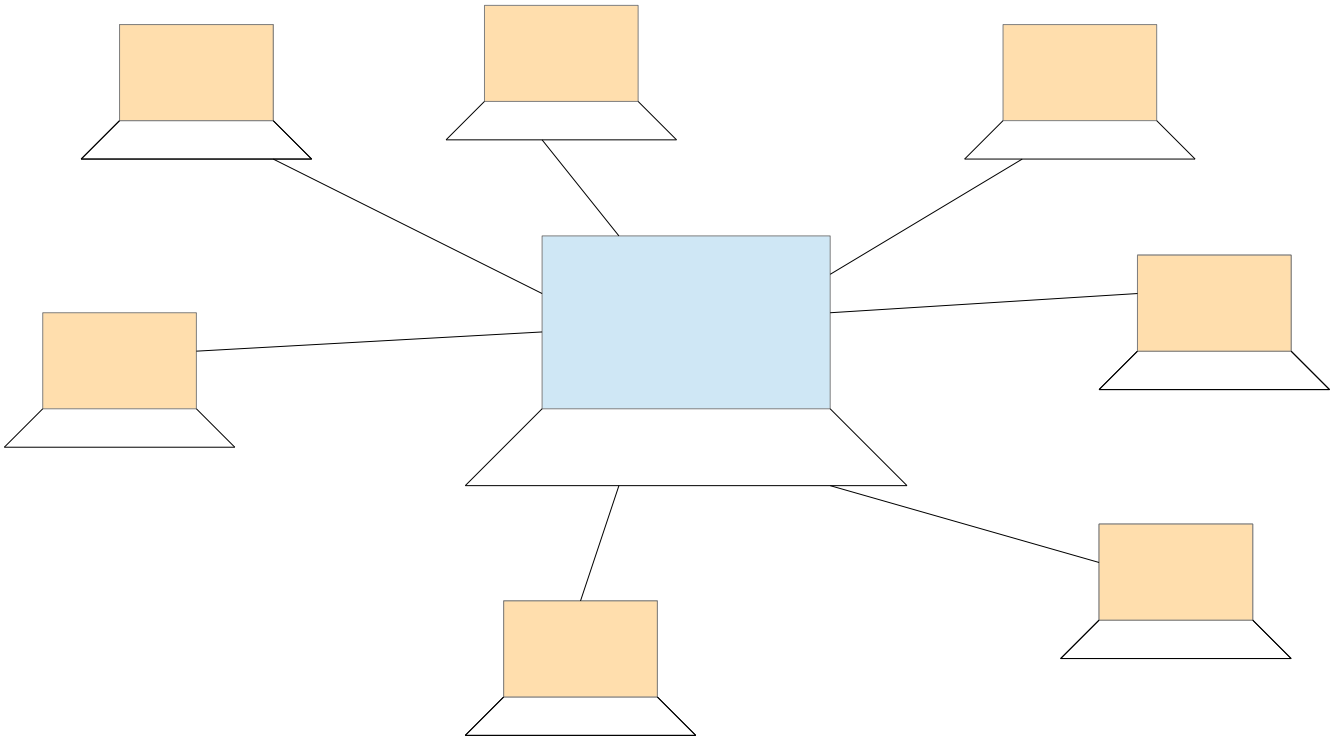
- Topologies (architecture):
  - Peer-to-peer – no hierarchy, shared nothing
    - Pastry, Chord, CAN systems
  - Hierarchical – one or more masters, several slaves
    - MapReduce based frameworks – Hadoop, SPARK, and many others

Peer-to-peer



Source: Google images

Hierarchical



Cloud

Source: Google images

P2P

- Every compute-node knows every other node.
- Every compute-node has a unique ID.
- Data gets distributed by some *function* $f(d_i)$ applied on each data item d_i
- The output of the $f(d_i)$ compared with node-id
- Data item sent to node whose ID is closest to $f(d_i)$

P2P

- How to decide the *function*?
  - Simplest is "distribted hash table" (DHT)
  - Uses a *hash* function.
- Compute-node IDs generated using the same function, using IP address or MAC etc as the data to be hashed.
  - It can even be random ID generation and allocation
- Data can be anything, text, graphs, tables.
  - Data item to hash on changes as per data type!

P2P (graphs)

- Decide data-unit to hash
  - A node? <hash(vertex-label), adj-vertices> (incoming or outgoing)?
  - An edge? <hash(s, p, o), s,p,o>
- How do you decide data-unit to hash?
  - Depends on the query types.
  - Why?

P2P (graphs)

- What do you join on?
  - Vertices or edges?
- So for distributed data what information do you need in one place?
  - Vertex labels?
  - Edge labels?
  - Why?

On each compute node

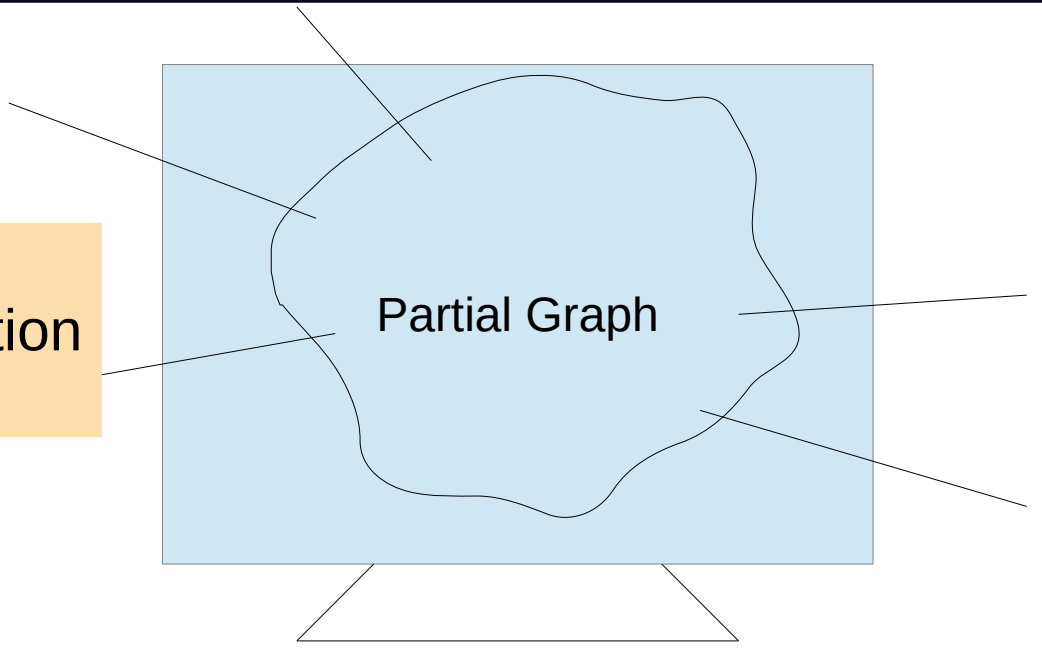
- There will be some graph vertices and their adjacent vertices
- For some graph vertices their adjacent vertices will not be on this compute-node.
- Then we will need to "*ship*" either this compute node's vertices, or get other compute node's vertices "*shipped*" to this compute-node.

Cost Consideration

- Pattern/join query cost computation remains the same.
- But *communication cost* gets added.
  - How to compute it?
- How to reduce communication cost?

Cost Consideration

How to ensure least communication between joins



How to reduce these "cuts"?

P2P DHT

- What entity to hash and distribute on?
  - What entities you join on? – Vertices!
- Hash vertex labels and decide their destination.
  - *<f(vertex-label), in-out-adj-list>*
  - Would this work?
  - Would any other strategy work?
  - Which would be low cost?

How to perform a join?

Shown on board…..