Data Intensive Computing - Review Questions 4

Deadline: Sep. 27, 2025

Group AKA
Ahmad Al Khateeb
Kusumastuti Cahyaningrum
Aleksandra Burdakova

1. **Compare Pregel's vertex-centric model with GraphLab's asynchronous computation. In which types of algorithms would one be preferable over the other?**
   Pregel's synchronous vertex-centric model is best for algorithms that naturally break into global supersteps, need strict phase separation, or benefit from simpler programming/debugging, e.g., PageRank, shortest paths, matching.

   GraphLab's asynchronous, dynamic model is better for ML/iterative algorithms that converge faster with fine-grained updates, require serializability (e.g., Gibbs sampling, ALS), or benefit from prioritized scheduling and reduced synchronization overhead, especially under load imbalance or cloud variability.

2. **Assume we have a graph and each vertex in the graph stores an integer value. Write three pseudo-codes, in Pregel, GraphLab, and PowerGraph to find the minimum value in the graph.**
   Pregel:

```
vertex.value = initial_int

function compute(messages):
  if superstep == 0:
    send_to_all_neighbors(vertex.value)
    vote_to_halt()
    return

  m = vertex.value
  for each msg in messages:
    m = min(m, msg)

  if m < vertex.value:
    vertex.value = m
    send_to_all_neighbors(m)

  vote_to_halt()
```

   Graphlab

```
// vertex.data holds current min
```

```
function update(v):
  old = v.data
  new = old
  for u in neighbors(v):
    new = min(new, u.data)

  if new < old:
    v.data = new
    for u in neighbors(v):
      schedule(u)   // re-run neighbors since min improved
```

PowerGraph

```
// vertex.data holds current min

GATHER(e = (u -> v)):
  return u.data

COMBINE(a, b):
  return min(a, b)

APPLY(v, gathered_min):
  old = v.data
  v.data = min(old, gathered_min)
  changed = (v.data < old)
  return changed

SCATTER(e = (v -> w), changed):
  if changed:
    signal(w)   // ensure w runs next round
```

3. **Assume we have two types of resources in the system, i.e., CPU and Memory. In total we have 28 CPU and 56GB RAM (e.g., 1 CPU = 2 GB). There are two users in the systems. User 1 needs ⟨2CPU, 2GB⟩ per task, and user 2 needs ⟨1CPU, 4GB⟩ per task. How do you share the resources fairly among these two users, considering (i) the asset fairness, and (ii) DRF.**

(i) Assume 1 Resource unit = 1 CPU (1 CPU = 2 GB)
User 1 → <2 CPU, 2GB> → 3 Resource unit
User 2 → <1 CPU, 4GB> → 3 Resource unit
asset fairness → $3x = 3y$
$\max(x,y)$
$2x + y \leq 28$
$2x + 4y \leq 56$
If task is not atomic:
User 1: $x = 9.33$:⟨66.7% CPU, 33.3%GB⟩($\sum$ =100%)
User 2: $y = 9.33$:⟨33.3% CPU, 66.7%GB⟩($\sum$ =100%)

If task is atomic:
User 1: x = 9:〈64% CPU, 32%GB〉(∑ = 96.4%)
User 2: y = 9:〈32%CPU, 64%GB〉(∑ = 96.4%)

(ii) Total resources: 〈28 CPU, 56GB〉

- User 1:〈2 CPU, 2GB〉: $\frac{2}{28}$ = 7.14% CPU and $\frac{2}{56}$ = 3.57% RAM;
  Dominant resource: CPU
- User 2:〈1 CPU, 4GB〉: $\frac{1}{28}$ CPU and $\frac{4}{56}$RAM;
  Dominant resource: RAM

max(x,y)
$2x + y \le 28$
$2x + 4y \le 56$
$\frac{2x}{28} = \frac{4y}{56} \rightarrow x = y$
If task is atomic:
User 1: x = 9:〈18% CPU, 18%GB〉
User 2: y = 9:〈9%CPU, 36%GB〉

4. **What are the similarities and differences among Mesos, YARN, and Borg in terms of architecture, scheduling, and fault tolerance?**
Scheduling:
Mesos → offer resources at framework level via resource offers (e.g., Hadoop, Spark).
Yarn → assign resources at task level
Borg → does scheduling at task level (finer granularity)

	Mesos	**YARN**	**Borg**
Architecture	Thin layer, two-level scheduling	Central RM + per-job AM	Central Borgmaster + Borglets
Scheduling	Resource offers (framework-driven)	Centralized RM, delegated to AMs	Feasibility + scoring (centralized)
Fault tolerance	ZooKeeper replication, framework-managed	RM HA, AM recovery limited	Paxos replication, automatic rescheduling

5. **Graph algorithms like PageRank define “importance” based on connectivity. From a critical perspective, what social voices or groups might be undervalued by such metrics? Suggest one alternative way to measure importance that includes context.**
Social voices or groups that might be undervalued by "importance" metrics in PageRank are for example marginalized voices (e.g., small activist communities, minority-language

publications), users who produce valuable but niche content, or non-mainstream networks where nodes don’t have dominant visibility.
One alternative way to measure importance would be contextual relevance, where importance is not only measured globally, but within specific communities of interest.

6. **Resource management systems usually aim for efficiency and fairness in technical terms. From an intersectional feminism lens, how might fairness be defined differently if we account for unequal starting positions of different users or communities?**
   From an intersectional feminism lens, fairness is not sameness, and equal allocation may still disadvantage communities with fewer resources, less historical access, or higher costs to participate. For example, a small research group may need more than an equal share to compensate for lack of historical computing capacity, compared to a tech giant with abundant alternatives.