PPIA

PPIA: A Peer-to-Peer Network for Smartphone CPU Power Sharing for Artificial Intelligence

Introduction:

PPIA is an innovative project that aims to harness the untapped computational power of smartphones worldwide to create a decentralized network capable of running large language models (LLMs) such as Llama-3, GPT-3, and GPT-4. By leveraging the collective processing power of these devices, PPIA offers a more environmentally friendly and cost-effective alternative to traditional data centers.

Smartphone Statistics:

According to Statista, there were approximately 6.6 billion smartphones in circulation worldwide in 2022. Assuming that 50% of these smartphones have a computational power between 0.1 and 0.5 teraflops, we can estimate that around 3.3 billion people possess smartphones with an average computational power of 0.3 teraflops. Multiplying this average by the number of people, we reach a staggering potential global computational power of nearly 1 exaflop.

Cluster Size Estimations:

To determine the number of smartphones required in a cluster to run different LLMs, we need to consider the computational power required for inference and the average power provided by each device.

Llama-3 (7 billion parameters):

- Estimated inference power: 0.03 teraflops
- Minimal cluster size: 1 smartphone
- Recommended cluster size for reliability: 10-20 smartphones

GPT-3 (175 billion parameters):

- Estimated inference power: 0.6 teraflops
- Minimal cluster size: 2 smartphones
- Recommended cluster size for reliability: 10-20 smartphones

GPT-4 (estimated up to 100 trillion parameters):

- Estimated inference power: 5-15 teraflops
- Minimal cluster size: 17-50 smartphones
- Recommended cluster size for reliability: 100-200 smartphones

Please note that these calculations are based on simplified assumptions and estimates. Actual performance may vary depending on factors such as network latency, distributed implementation efficiency, and variations in smartphone computational power.

Conclusion:
PPIA demonstrates the immense potential of harnessing the collective computational power of smartphones for running large language models. By creating a decentralized network, PPIA offers a more sustainable and accessible alternative to traditional data centers. However, further research and development are necessary to overcome challenges such as network latency, task distribution, and ensuring reliable performance across a wide range of devices.

K-means++ is an improvement over the standard K-means algorithm, particularly in the initialization of centroids. The main objective of K-means++ is to choose initial centroids that are far apart from each other, leading to better and more consistent clustering results. The algorithm follows these steps:

1. Choose the first centroid randomly from the dataset.
2. For each data point x, compute the distance d(x) between x and the nearest centroid that has already been chosen.
3. Choose the next centroid from the dataset with probability proportional to d(x)^2.
4. Repeat steps 2 and 3 until k centroids have been chosen.

5. Proceed with the standard K-means algorithm using the selected centroids.

The mathematical formulas involved in K-means++ are as follows:

1. Choosing the first centroid:
   - $LetX = x1, x2, ..., xn$ be the set of data points.
   - Choose the first centroid c1 uniformly at random from X.
2. Computing the distance to the nearest centroid:
   - For each data point x, compute the distance d(x) to the nearest centroid:

     $d(x) = mind(x, cj) : j = 1, 2, ..., i - 1$
     where d(x, cj) is the Euclidean distance between x and cj, and i-1 is the number of centroids already chosen.
3. Choosing the next centroid:
   - Choose the next centroid ci with probability proportional to d(x)^2:

     $P(ci = x) = d(x)^2 / \Sigma x \in X d(x)^2$
     where P(ci = x) is the probability of choosing x as the next centroid.
4. Repeat steps 2 and 3 until k centroids have been chosen.
5. Run the standard K-means algorithm using the chosen centroids:
   - Assign each data point to the nearest centroid.
   - Update the centroids by computing the mean of the data points assigned to each centroid.
   - Repeat the assignment and update steps until convergence or a maximum number of iterations is reached.

KMeans++ for PPIA algorithm logic:

The goal of this algorithm in our code architecture is to define the first parameter of user clusterisation:

1. Initialization:
    - Let $U = u1, u2, ..., un$ be the set of users, where each user ui is represented by a tuple (lat_i, lon_i, pcm_i), corresponding to the user's latitude, longitude, and PCM score, respectively.
    - Choose the first centroid c1 uniformly at random from U.
2. Computing the distance to the nearest centroid:
    - For each user u, compute the distance d(u) to the nearest centroid:

      $d(u) = mind(u, cj) : j = 1, 2, ..., i - 1$
      where d(u, cj) is a custom distance function that takes into account both the geographic distance and the PCM score difference between user u and centroid cj.
    - The custom distance function can be defined as:

      $d(u, cj) = sqrt(\alpha * geo_dist(u, cj)^2 + \beta * pcm_diff(u, cj)^2)$

    where:
        - $geo_dist(u, cj)$ is the geographic distance between user u and centroid cj, calculated using the Haversine formula or any other suitable method.
        - $pcm_diff(u, cj)$ is the absolute difference between the PCM scores of user u and centroid cj.
        - α and β are weight coefficients that determine the relative importance of geographic distance and PCM score difference in the overall distance calculation.
3. Choosing the next centroid:
    - Choose the next centroid ci with probability proportional to d(u)^2:

      $P(ci = u) = d(u)^2 / \Sigma u \in U d(u)^2$
    - where$P(ci = u)$ is the probability of choosing user u as the next centroid.
4. Repeat steps 2 and 3 until k centroids have been chosen.

5. Run the standard K-means algorithm using the chosen centroids:
    - Assign each user to the nearest centroid using the custom distance function.
    - Update the centroids by computing the mean latitude, longitude, and PCM score of the users assigned to each centroid.
    - Repeat the assignment and update steps until convergence or a maximum number of iterations is reached.

Pseudo-code for the adapted K-means++ algorithm:

```
function kmeans++(U, k, α, β):
    centroids ← initialize_centroids(U, k, α, β)
    while not converged:
        clusters ← assign_users(U, centroids, α, β)
        centroids ← update_centroids(clusters)
    return centroids, clusters

function initialize_centroids(U, k, α, β):
    centroids ← []
    centroids[0] ← random_choice(U)
    for i from 1 to k-1:
        distances ← compute_distances(U, centroids, α, β)
        probabilities ← distances^2 / sum(distances^2)
        centroids[i] ← random_choice(U, probabilities)
    return centroids

function compute_distances(U, centroids, α, β):
    distances ← []
    for u in U:
        min_dist ← infinity
        for cj in centroids:
            geo_dist ← geographic_distance(u, cj)
            pcm_diff ← abs(u.pcm - cj.pcm)
            dist ← sqrt(α * geo_dist^2 + β * pcm_diff^2)
```

```
            min_dist ← min(min_dist, dist)
        distances.append(min_dist)
    return distances
```

By incorporating both the geolocation and the PCM score into the distance calculation, the adapted K-means++ algorithm takes into account the geographical proximity and the computational power of users when forming clusters. The weight coefficients α and β allow you to adjust the importance of each factor according to your specific requirements.

Economic Viability:

Pricing Model:

- Option 1: Pay $15 per month to access various language models available on the PPIA network, such as Llama-3 and Mistral 7×8b.
- Option 2: Pay $15 per month and consent to contribute computational resources to power the network. In return, users have the opportunity to earn money and potentially cover their subscription cost or even generate profits.

Revenue Redistribution:

To incentivize users to participate in the network and contribute their computational resources, a portion of the revenue generated from subscription fees and usage by businesses should be redistributed to the users. A fair and attractive redistribution percentage would be 30% of the total revenue.

For example, if the total monthly revenue generated by PPIA is **$100,000**, then **$30,000** would be allocated for redistribution among the contributing users.

User Earnings:

The amount a user can earn monthly depends on several factors, such as the total revenue generated, the number of contributing users, and the individual user's

contribution to the network (determined by their PCM score and the tasks they complete).

Assuming a total monthly revenue of $100,000 and 30% redistribution to users, here's an estimation of the minimum and maximum monthly earnings per user:

- Minimum Monthly Earnings:
  If there are 100,000 contributing users and the redistribution is evenly split among them, the minimum monthly earning per user would be:
  ($30,000 / 100,000) = $0.30
- Maximum Monthly Earnings:
  If there are 10,000 contributing users and the top 1% of users (100 users) contribute significantly more than others, they could potentially earn a larger share of the redistribution pool. Assuming they collectively earn 20% of the redistributed revenue, the maximum monthly earning for a top contributor would be:
  ($30,000 * 0.20 / 100) = $60

Keep in mind that these are rough estimates, and actual earnings may vary based on the total revenue, the number of contributing users, and the distribution of computational power among users.

To make the system more equitable and rewarding, you can implement a tiered redistribution model based on users' PCM scores and contributions. For example:

- Tier 1 (Top 10% contributors): Receive 40% of the redistribution pool
- Tier 2 (Next 30% contributors): Receive 35% of the redistribution pool
- Tier 3 (Bottom 60% contributors): Receive 25% of the redistribution pool

This tiered approach ensures that users who contribute more to the network are rewarded proportionately, while still providing incentives for all participating users.

The economic viability of PPIA depends on several factors:

1. Attracting a large user base to contribute computational resources and generate revenue through subscriptions.
2. Ensuring that the revenue generated covers the operational costs of running the network, such as infrastructure, development, and maintenance.

3. Providing competitive pricing and incentives for businesses to use PPIA's services instead of traditional cloud computing providers.
4. Continuously improving the efficiency and performance of the network to minimize costs and maximize user earnings.

Exemple with 1Millions users, 10Millions users, 100Millions users:

Let's use the following thresholds to define the tiers:

- Tier 1: PCM ≥ 0.6 (top contributors, providing at least twice the average)
- Tier 2: 0.3 ≤ PCM < 0.6 (average contributors)
- Tier 3: PCM < 0.3 (below average contributors)

Now, let's calculate the distribution of users in each tier for the three cases, assuming the PCM score follows a normal distribution (bell curve).

Case 1: 1,000,000 users (60% free, i.e., 600,000 contributors)

- Tier 1 (15%): 600,000 * 0.15 = 90,000 contributors
- Tier 2 (35%): 600,000 * 0.35 = 210,000 contributors
- Tier 3 (50%): 600,000 * 0.50 = 300,000 contributors

Case 2: 10,000,000 users (60% free, i.e., 6,000,000 contributors)

- Tier 1 (15%): 6,000,000 * 0.15 = 900,000 contributors
- Tier 2 (35%): 6,000,000 * 0.35 = 2,100,000 contributors
- Tier 3 (50%): 6,000,000 * 0.50 = 3,000,000 contributors

Case 3: 100,000,000 users (60% free, i.e., 60,000,000 contributors)

- Tier 1 (15%): 60,000,000 * 0.15 = 9,000,000 contributors
- Tier 2 (35%): 60,000,000 * 0.35 = 21,000,000 contributors
- Tier 3 (50%): 60,000,000 * 0.50 = 30,000,000 contributors

Using these distributions, we can recalculate the average monthly revenue per contributor for each tier.

Case 1 (1,000,000 users):

- Tier 1: $5,300,000 * 0.40 / 90,000 = $23.56
- Tier 2: $5,300,000 * 0.35 / 210,000 = $8.83
- Tier 3: $5,300,000 * 0.25 / 300,000 = $4.42

Case 2 (10,000,000 users):

- Tier 1: $21,500,000 * 0.40 / 900,000 = $9.56
- Tier 2: $21,500,000 * 0.35 / 2,100,000 = $3.58
- Tier 3: $21,500,000 * 0.25 / 3,000,000 = $1.79

Case 3 (100,000,000 users):

- Tier 1: $183,500,000 * 0.40 / 9,000,000 = $8.16
- Tier 2: $183,500,000 * 0.35 / 21,000,000 = $3.06
- Tier 3: $183,500,000 * 0.25 / 30,000,000 = $1.53

Ecological benefit:

Assumptions:

- Average PCM score: 0.3
- Average computational power per user: 0.3 teraflops
- Power consumption of a user's device while contributing: 5 watts

- Power Usage Effectiveness (PUE) of a data center: 1.5
- CO2 emissions per kWh: 0.5 kg (global average)

Computational Power:

- 1 million users: 1,000,000 * 0.3 = 300,000 teraflops
- 10 million users: 10,000,000 * 0.3 = 3,000,000 teraflops
- 100 million users: 100,000,000 * 0.3 = 30,000,000 teraflops
- 1 billion users: 1,000,000,000 * 0.3 = 300,000,000 teraflops

Electrical Energy:

Assuming users contribute 50% of the time (12 hours per day):

- 1 million users: 1,000,000 * 5W * 12h * 365 days = 21,900,000 kWh/year
- 10 million users: 219,000,000 kWh/year
- 100 million users: 2,190,000,000 kWh/year
- 1 billion users: 21,900,000,000 kWh/year

CO2 Emissions Saved:

Data center emissions = User emissions * PUE
CO2 savings = Data center emissions - User emissions

- 1 million users:
  - User emissions: 21,900,000 kWh * 0.5 kg/kWh = 10,950,000 kg CO2
  - Data center emissions: 10,950,000 kg * 1.5 = 16,425,000 kg CO2
  - CO2 savings: 16,425,000 kg - 10,950,000 kg = 5,475,000 kg CO2

- 10 million users:
  - CO2 savings: 54,750,000 kg CO2
- 100 million users:
  - CO2 savings: 547,500,000 kg CO2
- 1 billion users:
  - CO2 savings: 5,475,000,000 kg CO2

These estimates suggest that by harnessing the computational power of users' devices, PPIA could potentially save a significant amount of CO2 emissions compared to traditional data centers. However, it's essential to keep in mind that these calculations are based on simplified assumptions and may not reflect the full complexity of the system.

Other factors to consider include:

- Variability in user contribution time and device power consumption
- Efficiency improvements in data centers over time
- Differences in regional CO2 emissions per kWh
- Embodied carbon in the production and disposal of user devices

Despite these limitations, the estimates provide a rough indication of the potential environmental benefits of PPIA's decentralized approach to AI computing. As the platform grows and collects real-world data, more accurate assessments of its environmental impact can be made.