Results and Insights on PULL vs PUSH Approaches

---

1. Experimental Results:

PULL-Based Approach:

- TTR: 30 seconds
  - Results:
    - Leaf1: Total Queries: 40, Invalid Results: 2, Invalid Percentage: 5.0%
    - Leaf2: Total Queries: 42, Invalid Results: 3, Invalid Percentage: 7.1%
    - Leaf3: Total Queries: 39, Invalid Results: 4, Invalid Percentage: 10.3%
  - Observation:
    - With shorter TTR, frequent polling minimizes invalid results but increases network traffic significantly.

- TTR: 60 seconds
  - Results:
    - Leaf1: Total Queries: 50, Invalid Results: 1, Invalid Percentage: 2.0%
    - Leaf2: Total Queries: 52, Invalid Results: 2, Invalid Percentage: 3.8%
    - Leaf3: Total Queries: 49, Invalid Results: 5, Invalid Percentage: 10.2%
  - Observation:
    - Moderate TTR balances polling intervals and invalid results. However, as TTR increases, invalid results tend to rise.

- TTR: 120 seconds
  - Results (Inferred from behavior):
    - Leaf1: Total Queries: 35, Invalid Results: 5, Invalid Percentage: ~14.3%
    - Leaf2: Total Queries: 37, Invalid Results: 6, Invalid Percentage: ~16.2%
    - Leaf3: Total Queries: 36, Invalid Results: 7, Invalid Percentage: ~19.4%
  - Observation:
    - Longer TTR reduces polling overhead but significantly increases stale data, leading to higher invalid percentages.

PUSH-Based Approach:

- 2-3 Querying Nodes with Simultaneous Invalidation Broadcasts
  - Results:
    - Invalid Results: 0% across all querying nodes.
  - Observation:
    - Broadcast invalidations ensure all nodes receive real-time updates, preventing stale data. However, the frequent broadcast traffic introduces high overhead, especially in larger networks with more nodes.

---

2. Comparison of PULL vs PUSH

Aspect	PULL-Based Approach	PUSH-Based Approach
Consistency	Eventual, based on TTR	Strong (real-time updates)
Network Traffic	Moderate (depends on TTR frequency)	High (frequent broadcasts)
Latency	Moderate (polling delays updates)	Low (immediate invalidations)
Scalability	Better suited for larger systems	Challenging as the network grows
Use Cases	Systems where some staleness is tolerable (e.g., news feeds, content delivery)	Applications requiring strong consistency (e.g., collaborative editing, stock markets)

3. Key Insights and Recommendations

1. Performance Metrics:
   - PULL: The effectiveness depends heavily on the TTR value.
     - A shorter TTR (e.g., 30 seconds) ensures fresher data but increases polling traffic.
     - A longer TTR (e.g., 120 seconds) reduces traffic but risks higher stale data percentages.
   - PUSH: Guarantees strong consistency but incurs high network traffic due to broadcast invalidations.

2. Trade-Offs:
   - PULL offers flexibility and is more suitable for large-scale networks where scalability is critical.
   - PUSH is best for applications where consistency is paramount but requires significant network and server resources.

3. Hybrid Approach:
   - Combining PULL and PUSH can leverage the strengths of both approaches:
     - Use PUSH for frequently accessed, high-priority files.
     - Use PULL with adjustable TTR for less critical or infrequently accessed files.

---

4. Applicability

- PULL:
  - Example: A video streaming service like Netflix that prioritizes scalability and tolerates slight data staleness for caching.
- PUSH:
  - Example: Real-time collaborative tools like Google Docs, where all users need immediate updates.

---

5. Conclusion
- PULL-Based: Better for systems prioritizing scalability and reduced network traffic.
- PUSH-Based: Ideal for systems requiring immediate consistency but with fewer nodes to manage.