SWOT Analysis
RAFT Consensus algorithm

Strengths:

- *Easy to understand and implement:*

  RAFT is designed to be easy to understand and implement, unlike other distributed system algorithms preceding it like Paxos[1].
- *Ensures availability, recovers from failures quickly:*

  The system ensures recovery in case of leader failure by evoking a speedy leader election. It tolerates up to (n-1)/2 server failures, where n is the total number of servers in a cluster.

  Practical analysis:

  The following steps were taken to test availability on a RAFT system of 3 nodes:
  1) Run a command while all nodes are alive
  2) Suspend leader, Run a command and check if response is proper.

  Results: System was available in both cases, properly responding despite node's failure.
- *Ensures consistency:*

  Log replication ensures consistency of the sequence of operations performed across all nodes, even after failure and reactivation of nodes in the system cluster.

  Practical analysis:

  The following steps were taken to test consistency on a RAFT system with 3 server nodes:
  1) Setting the value of X while all nodes are alive
  2) Getting the value of X to ensure that it has been written
  3) Suspending node 0 (node failure), then setting the value of y
  4) Getting the value of y to ensure that it has been written
  5) Reactivating Node 0, then getting the value of y from the reactivated node to ensure that it has the latest write

  Results: despite Node 0 failing and getting reactivated, the system was able to propagate the missing information committed during its failure time and ensure consistency throughout the system.

Weaknesses:

- *Lack of partition tolerance:*

  In case of a partition, the system continues to operate with the leader of the partition of most nodes. If the system has no such partition, the system may fail as it does not handle byzantine faults [3].
- *Reduced throughput and latency in heavily loaded systems*

  Leader acts as a bottleneck to all processes and reduces throughput in heavily loaded systems since it performs n log replications [4].

- *Complex to configure for large scale systems*
  The performance of the RAFT consensus algorithm becomes highly inefficient in large scale systems, as it is not configurable and performs many log replication operations[5].

Opportunities:

- *Can be integrated with other distributed systems' technologies*
  RAFT is designed to be modular, which means it can be used in different parts of a distributed system with different requirements[6].
- *Can be used in various applications*
  RAFT's simplicity allows it to be easily extendable for specific use cases; it has helped distributed systems be more easily applicable in practice[6].
- *RAFT's simplicity makes it easily applicable.*
  There's a growing need in the industry for distributed systems, which calls for easily implemented algorithms like RAFT.[7]

Threats:

- *Competition from other consensus algorithms*
  Other consensus algorithms are already well established in the industry like Paxos and ZAB. They could be preferred over RAFT[8].
- *Risk of improper implementation*
  RAFT's simplicity is a double edged sword. Although it is simple, it can lead to developers underestimating the importance of every step, which might lead to an improper implementation resulting in avoidable bugs such as data loss and inconsistency.
- *Risk of becoming obsolete*
  Competing consensus algorithms may be developed to address some of the weaknesses of RAFT, potentially making it obsolete[8].

References:

[1] Ongaro, D. and Ousterhout, J., 2014. In search of an understandable consensus algorithm. In 2014 USENIX Annual Technical Conference (USENIX ATC 14) (pp. 305-319). USENIX Association.
[2] Ongaro, D. and Ousterhout, J., 2015. Consensus: Bridging theory and practice. ACM Queue, 13(7), p.44.
[3] https://ilyasergey.net/CS6213/week-03-bft.html
Kafura, D. and Lee, S.H., 2015. A survey of consensus protocols in distributed systems. ACM Computing Surveys (CSUR), 47(1), pp.1-36.
[4] Junqueira, F.P., Reed, B.C. and Serafini, M., 2011. Zab: High-performance broadcast for primary-backup systems. In Distributed Computing (pp. 418-431). Springer, Berlin, Heidelberg.
[5] https://www.cncf.io/blog/2019/11/04/building-a-large-scale-distributed-storage-system-based-on-raft/
[6] Ongaro, D. and Ousterhout, J., 2015. Consensus: Bridging theory and practice. ACM Queue, 13(7), p.44.
[7] Ongaro, D. and Ousterhout, J., 2014. In search of an understandable consensus algorithm. In 2014 USENIX Annual Technical Conference (USENIX ATC 14) (pp. 305-319). USENIX Association
[8] Kafura, D. and Lee, S.H., 2015. A survey of consensus protocols in distributed systems. ACM Computing Surveys (CSUR), 47(1), pp.1-36.