Performance Considerations

in a DLT/Blockchain World

|  |  |  |
| --- | --- | --- |
| Change Doc Log | | |
| Date | Change | By |
| 10/15/2017 | Draft 1 | Guy Hochstetler |
|  |  |  |

Note: if there’s a wiki to maintain this, we can forego the change doc log.

* Decide **what are we measuring?**
  + Throughput – how do we define a “transaction” in DLT?

“Transaction” in the context of Distributed Ledger Technology (DLT) aka “Blockchain” is an activity that, once final, transitions one or more object/asset states collectively, and atomically, from current state (which could be “none” if the asset does not yet exist) to future state.

* + Memory consumption
  + Storage consumption
  + CPU cycles
  + What are the key attributes to measure for blockchain vs those that are part of other layers? E.g., this isn’t an integration layer so should we be concerned with data parsing? Serialize and de-serialize?
* **What are the boundaries** of what we decided to measure?

The Transaction performance boundary, for measurement purposes, begins from the transaction submission point, i.e., submission to the DLT platform for commitment, to its finality within the DLT network, however that is achieved, however that is achieved by the platform.

* + Where are the hand-off points?
  + Are they standardized?
  + How do you isolate the boundary points?
* What are the appropriate performance metrics for the solution? What is the use case being measure? There are always trade-off compromises for performance where, for example security is critical.
  + If long running tasks are part of the solution are there appropriate timeout settings?
  + Human involvement?
  + Complex transactions, e.g., update of more than one state atomically?
  + How is security valued versus performance?
* What are the dependent components?
  + what database is used? Note that databases have a standard by which transactions are defined and measured (see [TPC](http://www.tpc.org/))
  + What is the makeup of the TCP/IP network layer across peers in the DLT network?
  + Is there data persistence? Is it optional/configurable?
  + What messaging protocol is used between nodes?
* What are the dependent configurations?
  + Which Consensus model is employed in the test?
    - Which algorithm is utilized, e.g., PoW, PoS, PBFT, RAFT
    - Validating or Non-validating?
    - Is there a separate notion of validating: 1) uniqueness to protect from double-spending; and 2) chain-of custody to assure valid ownership?
  + Network settings across all peers? Where are the known bottlenecks?
  + Nodes:
    - How many nodes in the network?
    - How many nodes involved in transaction?
    - How many are read-only nodes?
    - How many nodes are involved in validating?
  + What is the complexity of the contract?
  + What is the size of the asset/state object included in the transaction? Is there more than one if a complex atomic transaction?
* How do you measure the key items?
  + What tools are selected?
  + What data are being captured?
  + Where are you storing the results?
  + How are the results validated?
* Is the performance test environment appropriately reflecting the desired conditions?
  + Number and speed of processors?
  + Message/object size and complexity?
* Implications of processing pattern choices. This is tied to use cases.
* What component characteristics affect the performance? What are the potential tunable attributes?