**Classifications**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Type** | **(Unique, Only One)** | **Description** |
| t-bug | Bug-related issues | The issue is about existing code that is buggy. A bug by definition is a wrong design/implementation that changes the semantic of program. |
| t-new | New design/feature | The issue is about a complete proposal to new design/features that are NOT low priority. We only consider the new designs that the **developers** really suggest. |
| t-low | Low priority, will be filtered out issues | Dangling issue (not an issue, developers do not care), subtask, duplicate, and invalid issue. The issue is about existing code, but just additional maintenance, compatibility issues, cleaning up code, and adds more debugging code. |
| ***(If the bug type is t-low, we DO NOT add more tags)*** | | |
|  |  |  |
|  |  |  |
| **Aspects** | **(Can be Multiple)** | **Description** |
| a-perf | About performance | Performance degrades |
| a-reli | About reliability | Ex: data loss, data corruption, operations/jobs get error/unfinished |
| a-avail | About availability | Ex: node down, cluster down, hang |
| a-cons | About consistency | Ex: replicas are not consistent, data center/geo-replica/eventual/causal-consistency issue, |
| a-prog | About programmability | Ex: add API, etc. (this is about user programmability, NOT software design bugs) |
| a-qos | About Quality of Service | In QoS issue, overall performance is fine, but when the cluster is shared across multi-users/tenants, people don’t get fairness. |
| a-scale | About scalability | Ex: the system works fine on 10 nodes, but not in 100 nodes. Elasticity issue is also about scalability. Or the program works fine when dealing with small data, but when dealing with load spikes or large number of data, the code is not scalable. |
| a-sec | About security |  |
| a-topo | About topology | Ex: the system works on 2 racks, but not on 4 racks, or in 1 DC, but not multiple DC. |
|  |  |  |
|  |  |  |
| **HW Fault** | **(Can be multiple)** | **Description** |
| hw-core | Processor/core issue | Processor/core is problematic |
| hw-disk | Disk issue | Disk failure, disk corruption, flaky disks, disk is full |
| hw-mem | Memory issue | Memory corruption, memory failure (can’t read/write to memory) |
| hw-net | Network issue | Network disconnection, flaky network |
| hw-node | Machine/node issue | Machine/node failures. The issue does not describe what particular HW is causing the issue. |
|  |  |  |
|  |  |  |
| **HW Fault Type** | **(Can be multiple)** | **Description** |
| ***(If the issue is hardware fault, we tag the fault type)*** | | |
| ht-corrupt | Corrupt data | The type of failure is corrupt data (coming from disk), or perhaps corrupt input from users |
| ht-limp | HW just limps | HW consistently has performance degradation. |
| ht-stop | Fail-Stop Failure | The type of hardware failure is fail stop, network disconnected, core dies, and disk dies. |
|  |  |  |
|  |  |  |
| **SW fault type** | **(Can be Multiple)** | **Description** |
| sw-config | Config issue | Users enter wrong configuration, or external configuration such as OS configuration is wrong. Or, the system does not handle all possible configurations. |
| sw-eh | Error handling, error code or exception handling, or fail silent | The software does not catch a new error code or exceptions that can be thrown by the software itself, the OS, or other components. |
| sw-hang | Hang | Could be because of deadlock, or other causes (infinite loop, etc.) |
| sw-load | Load | User load / background load / peak load that is unexpected |
| sw-logic | Incorrect logic | Usually wrong control flow, or wrong computation (it’s like a logic bug) |
| sw-opt | Optimization issue | The code is not optimized, so they optimize the data structure, for performance, memory efficiency, etc. |
| sw-race | Data race (internal concurrency) | Thread re-ordering execution, internal non-determinism that is not handled properly, that leads to data races! (Not deadlock!). Network reordering, external non-determinism that is not handled properly can also lead to race. |
| sw-space | Space / resource issue | E.g. no disk space, or small disk space, or out of memory, and the system cannot handle this, or doesn’t do a good of space management. No resource. |
|  |  |  |
|  |  |  |
| **Implications** | **(Can be Multiple)** | **Description** |
| i-loss | Data loss |  |
| i-corrupt | Data corruption |  |
| i-down | System down (not available) | This could be system crash, hang, deadlock |
| i-opfail | Operation gets an error failure | Ex: Operation is not finished, returns an error. Job is not finished, etc. |
| i-stale | Inconsistency issues | Some data replicas are inconsistent/stale (it’s about consistency issue) when they’re supposedly consistent. |
| i-perf-?x | Performance failure. “-?x” is optional. | Performance failures. i-perf-2x means the system slows down by 2x. |
|  |  |  |
| **Scale of issue** | **(Unique, Only One)** | **Description** |
| (Default none) |  | It means either it only affects 1 machine, or it is implicit how many are affected. |
| x-m | About multiple machines | E.g. multiple machines are down, or the impact affects multiple machines. |
| x-c | About whole cluster | E.g. whole cluster is affected greatly. |
|  |  |  |
|  |  |  |
| **Component** | **(Can be multiple)** |  |
| c-xyz | Xyz is a sub-component of the system that is **buggy** in the issue. |  |
| **Mapreduce** | | |
| c-cli | Client code | Example library, etc. |
| c-boot | Boot process |  |
| c-map | Map phase |  |
| c-red | Reduce phase |  |
| c-shuf | Shuffle phase |  |
| c-comm | Commit phase |  |
| c-sx | Speculative execution |  |
| c-tt | Task tracker |  |
| c-jt | Job tracker |  |
| c-nm | Node manager |  |
| c-am | Application master |  |
| c-rm | Resource Manager |  |
| c-sc | Scheduler algorithm |  |
| c-ipc | IPC/RPC protocol |  |
| c-sec | Security |  |
| c-logj | Output logging |  |
| c-hs | Application history server |  |
| c-fs | Other storage stuff e.g hdfs |  |
| c-dtcp | Distcp/ distributed copy |  |
| c-etc | Things incluce tests, stream, etc |  |
| **HDFS** | | |
| c-boot | Boot process |  |
| c-dn | Datanode |  |
| c-nn | Namenode |  |
| c-cli | Client library |  |
| c-ss | Snapshot file system (name space) |  |
| c-ha | High availability (namenode failover) |  |
| c-jrnl | Journaling |  |
| c-read | Data/metadata, including caching |  |
| c-write | Data/metadata write, hardlink, truncate, etc. |  |
| c-raid | HDFS raid |  |
| c-rpc | RPC |  |
| c-fsck | Background fsck check |  |
| c-rep | Replication, policy, data placement policy |  |
| c-proxy | NFS gateway, HTTP gateway |  |
| c-etc | all other components/protocols not part of all of the above | Example: JNI, C API, REST API, network interface, etc. |
| HBase | | |
| c-boot | Boot process |  |
| c-comp | Compaction |  |
| c-rsv | Region server |  |
| c-mas | Master |  |
| c-cli | Client |  |
| c-read | Read from memstore |  |
| c-write | Write to memstore |  |
| c-log | Write to commitlog |  |
| c-flush | Flushmemstore to sstables |  |
| c-rsp | Region split |  |
| c-lsp | Log splitting, handle dead region server |  |
| c-fsck | Data check, consistency check |  |
| c-zk | Zookeeper related |  |
| c-cop | Coprocessors |  |
| c-ipc | IPC layer |  |
| c-snap | Snapshot |  |
| c-ns | Name space |  |
| c-fs | File system related |  |
| c-cross | Other related component |  |
| **Cassandra** | | |
| c-boot | Bootstrapping process (might involve c-str too) |  |
| c-clean | Process of removing deleted data. | i.e. tombstone |
| c-cli | Command Line Interface | Client interface to cassandra, e.g. cql, thrift, cqlsh, transport, jmx, nodetool |
| c-clog | Commit log |  |
| C-comp | Compaction of SSTable | e.g. the process of merging SSTables to reduce disk usage |
| c-cross | Connection with another system. | e.g. hadoop (via Pig), etc |
| c-ent | Anti-Entropy (aka Manual Repair) (reduce inconsistency) |  |
| c-get | Get Range Slice & Read Operation, including read repair |  |
| c-gms | Gossiper (P2P comm protocol) |  |
| c-hint | Hinted Handoff | Missed writes that stored in other replicas, reduce inconsistency. |
| c-io | Cassandra IO component | These are parts of their 'core' code, e.g. compression, disk writer, network socket management. |
| c-mem | Memtable, Caching |  |
| c-mig | Migration | Decommissioning nodes |
| c-mut | Mutate Operation | Insert, Delete, Paxos, etc. |
| c-part | Partitioner | Data distribution method across the nodes), Virtual node (auto-balance the data across the cluster e.g. one node might responsible for multiple ranges. |
| c-snitch | Snitch | Topology info or locator component |
| c-sst | sstable | E.g. sstable row indexes.  This includes counters e.g. column counter and bloom filter. |
| c-str | Streaming | Usually triggered by bootstrapping or decommissioning nodes, or related with messaging inter node. |
| c-etc | security module (all authentication module) |  |
| **Zookeeper** | | |
| c-le | Leader Election |  |
| c-zab | ZooKeeper Atomic Broadcast |  |
| c-ss | Snapshot |  |
| c-cli | Client library code |  |
| **Flume** | | |
| c-src | Source (Flume OG and Flume NG) |  |
| c-chan | Channel (Flume NG) |  |
| c-sink | Sink (Flume OG and Flume NG) |  |
| c-cprov | Specifying configuration from properties. |  |
| c-mas | Flume OG master node | This tag is used for lifecycle supervisor in Flume NG. |
| c-col | Flume OG logical node | Usually it acts as collector or agents. Decorator in Flume OG also included in this tag. |