Load/consistency test design

Environment setup

N servers nodes, C client nodes, each client node should be assigned a unique sequence number n. Data nodes should be restarted continuously (not more than B nodes at a time where B is number of backups configured for the cache).

Test cases

Stability and availability

All tests should not experience any hangs during server nodes restarts.

Implicit operation automatic retries

Test should use implicit cache operations without catch-blocks. Each client should generate continuous load to the cluster (random get, put, invoke, remove operations). No exceptions must be thrown to the user code.

Transactional write/read consistency

Each client generates a random integer K in a limited range (say, [0, 100000]) and creates 5 (may be configured) keys in the form 'key-' + K + '-1', 'key-' + K + '-2', ... Then client starts a pessimistic repeatable read transaction, reads value associated with each key. Values must be equal. Client increments value by 1, commits the transaction. Client should retry in the case of topology exceptions possibly thrown from cache operations.

Transactional write/invoke consistency

Each client generates a random integer K in a limited range (say, [0, 100000]) and creates 5 (may be configured) keys in the form 'key-' + K + 'master', 'key-' + K + '-1', 'key-' + K + '-2', ... Then client starts a pessimistic repeatable read transaction and randomly chooses between read and write scenarios:

- Reads value associated with the master key and child keys. Values must be equal.

- Reads value associated with the master key, increments it by 1 and puts the value, then invokes increment closure on child keys. No validation is performed.

Transactional invoke retry consistency

Each client generates a random integer K in a limited range (say, [0, 100000]) and creates a batch of 5 (may be configured) keys in the form 'key-' + K + '-' + n + '-1', 'key-' + K + '-' + n + '-2', ... where n is a unique sequence number assigned to the client. Each client maintains a local map that it updates together with cache. Client invokes an increment closure for all generated keys (no explicit transaction is used) and atomically increments value for corresponding keys in the local map. No exceptions must be thrown to the user code. To validate cache contents, all writes from the client should be stopped, values in the local map must be equal to the values in the cache.