Università degli Studi di Roma Tor Vergata
Corso di Laurea Magistrale in Ingegneria Informatica
Ingegneria del Software II (9 CFU) A.A. 2020/2021

Report: identificazione di partizioni di dominio e boundary analysis per i metodi `createLedger` della classe `BookKeeper` nel progetto open source "BookKeeper"

Massimo Stanzione, matr. 0304936

Si considerano i metodi `createLedger` della classe `BookKeeper`, le cui segnature sono di seguito riportate:

M1. `createLedger(DigestType digestType, byte[] passwd)`
M2. `createLedger(int ensSize, int qSize, DigestType digestType, byte[] passwd)`
M3. `createLedger(int ensSize, int writeQuorumSize, int ackQuorumSize, DigestType digestType, byte[] passwd)`
M4. `createLedger(int ensSize, int writeQuorumSize, int ackQuorumSize, DigestType digestType, byte[] passwd, Map<String, byte[]> customMetadata)`

Nella loro implementazione, i metodi M1, M2 ed M3 effettuano una chiamata al metodo M4 (overloading). Tutti i metodi sono di tipo `LedgerHandle`.

Identificazione dei domini di input

Si è proceduto anzitutto alla identificazione dei domini di input, ed in particolare si è fatto riferimento alla documentazione del protocollo BookKeeper per individuare i concetti chiave, ottenendo così il significato di **Ledger**, oggetto di interesse dei metodi in esame, descrivibile come sequenza di record ("entries") ciascuno dei quali contenente metadati e riferimenti ai seguenti concetti:

- **Bookie**: nodo (server) nel quale un Ledger può essere memorizzato;
- **Ensemble**: gruppo di Bookies selezionati per la memorizzazione di ciascuna entry di un Ledger; ogni Ledger ha un numero di Ensemble possibili E;
- **Quorum di scrittura** Qw: numero di nodi nei quali i record vengono scritti
- **Quorum di acknowledgement** Qa: numero di nodi che riconoscono (acknowledgement) una scrittura, sottoinsieme di Qw.

Ulteriore identificazione è fornita dai dettagli implementativi: il progetto è scritto in linguaggio Java, con riferimenti ai tipi di dato elementari `int` e `byte`, ai tipi `String` e `Map` e ad una enumerazione, `DigestType`, dichiarata come segue (*rr. 670-686*):

```
public enum DigestType {
    MAC, CRC32, CRC32C, DUMMY;
    //…
    }
```

Sono inoltre presenti condizioni imposte come vincoli di integrità: dalla documentazione BookKeeper è infatti esplicitamente posta la condizione

$$\mathbf{E} \geq \mathbf{Qw} \geq \mathbf{Qa}$$

ed è inoltre specificato il comportamento che il sistema assume nel caso in cui tale condizione venisse violata:

> *"the ensemble size (E) must be larger than the write quorum size (Qw), which must in turn be larger than the ack quorum size (Qa). If that condition does not hold, then the ledger creation operation will fail.".*

Dalla documentazione si ricava inoltre che $\mathbf{Qa} \geq \mathbf{1}$: infatti, oltre ad essere ricavabile dal contesto,

> *"The system can tolerate $Qa - 1$ failures without data loss."*

Identificazione delle classi di equivalenza

Una volta individuati i domini di input si può procedere alle definizioni di classi di equivalenza per i parametri in analisi. A tale scopo si possono individuare i seguenti criteri:

C1. **Contesto del dominio di input**
Vedasi paragrafo precedente

C2. **Condizioni dichiarate/ricavate in documentazione** $\{\mathrm{E} \geq \mathrm{Qw} \geq \mathrm{Qa};\ \mathrm{Qa} \geq 1\}$
Esplicitamente definita nella documentazione di BookKeeper

C3. **Valore nullo**
Per enumerazioni, stringhe e vettori, test sul comportamento del sistema

C4. **Definizione esplicita di tutti i valori ammissibili all'interno di una enumerazione**
Si considera una CdE differente per ogni valore assegnato all'enumerazione

C5. **Dettagli implementativi in linguaggio Java**
Valori assumibili dallo specifico tipo di dato in cui il parametro è stato implementato in linguaggio Java, secondo la documentazione ([-2^{31}, 2^{31}-1] per `int`, [-128, 127 per `byte`])

C6. **Confronto con i valori di default assegati nell'implementazione**
`ensSize=3; qSize=writeQuorumSize=ackQuorumSize=2` (rr. 851,870)

Parametro	Classi di equivalenza	Criteri
`qSize`	$\{\leq 0\}$; $\{\in [1, 2^{31}\text{-}1]\}$; $\{\geq 2^{31}\}$	C1, C5, C6
`ackQuorumSize`	$\{\leq 0\}$; $\{\in [1, 2^{31}\text{-}1]\}$; $\{\geq 2^{31}\}$	C1, C5, C6
`writeQuorumSize`	{< `ackQuorumSize`}; {≥ `ackQuorumSize`}	C2
`ensSize`	Metodo M2 {< `qSize`}; {≥ `qSize`} Metodi M3, M4 {< `writeQuorumSize`}; {≥ `writeQuorumSize`}	C2
`digestType`	{`"MAC"`}; {`"CRC32"`}; {`"CRC32C"`}; {`"DUMMY"`}; {`null`}	C3, C4
`passwd`	{`null`}; {array vuoto}; {array aventi valori fuori dal range di byte}; {array aventi valori entro il range di byte}	C3, C5
`customMetadata`	Per il parametro di tipo `String`: {`null`}, {non-`null`} Per il parametro di tipo `byte`[]: come `passwd` Inoltre, `null`.	C3, C5

Nella definizione delle CdE sono state effettuate le seguenti semplificazioni e considerazioni:

- `ackQuorumSize`: per i numeri negativi non viene considerato il limite minimo -2^{31} in quanto ci si aspetta che il sistema risponda allo stesso modo sia per numeri minori che maggiori di tale limite, poiché un numero nullo o negativo di server che riconoscono una scrittura produce sempre una situazione di errore;
- `passwd`: non è presente alcun vincolo sulla lunghezza dell'array di byte, quindi nessun limite superiore è stato considerato.
- `digestType`: si assume che il valore sia già appartenente alla enumerazione, o al più nullo.

Boundary analysis

Nella scelta di valori rappresentativi per le CdE si può fare riferimento a valori ai "confini" delle singole classi, in quanto più proni a produrre errori. Nel dettaglio:

- `ackQuorumSize`, `qSize`: 0; 1; 2^{31}-1; 2^{31}
- `writeQuorumSize`: `ackQuorumSize`-1; `ackQuorumSize`; *(`ackQuorumSize`+1)*
- `ensSize`: `writeQuorumSize`-1; `writeQuorumSize`; *(`writeQuorumSize`+1)*
- `passwd`: array di lunghezza arbitraria (non specificata) aventi
  - almeno un byte di valore minore di -128
  - byte con valori compresi tra -128 e 127
  - almeno un byte con valore maggiore di 127

Possibili test da includere e risultati attesi

Seguendo un approccio di tipo unidimensionale, una suite di test possibile potrebbe prevedere chiamate ai metodi M2 ed M4, selezionando come parametri di input i valori ottenuti dalla boundary analysis in modo indipendente (quindi senza effettuare il prodotto cartesiano tra le CdE) fino ad includere nei test tutte le CdE.

I risultati attesi dai test possono essere definiti come una istanza di `LedgerHandle` nei casi in cui si assume che i parametri in input fornisca un comportamento del sistema conforme alle specifiche, mentre si può considerare un sollevamento di eccezione (`BKNotEnoughBookiesException` oppure `IncorrectParameterException`, entrambe nella classe `LedgerCreateOp` chiamata da M4) in caso contrario.