

INGEGNERIA DEL SOFTWARE 2: REPORT TESTING.**Sommario**

METODOLOGIA	2
FASTJSON	2
BOOKKEEPER	3
IDENTIFICAZIONE PARTIZIONI DI DOMINIO	3
BOUNDARY ANALYSIS	4
IDENTIFICAZIONE CASI DI TEST	5
ZOOKEEPER	11
IDENTIFICAZIONE PARTIZIONI DI DOMINIO	11
BOUNDARY ANALYSIS	12
IDENTIFICAZIONE CASI DI TEST	13
PROFILI OPERAZIONALI	18
BOOKKEEPER	18
ZOOKEEPER	18
LINKS	18
FIGURE	19

METODOLOGIA.

Le tecniche e gli strumenti descritti durante il corso sono stati sperimentati sui tre progetti **Apache BookKeeper**, **Apache Zookeeper** e **Fastjson**. Per i progetti Apache BookKeeper e Apache Zookeeper, ho eseguito il fork del repository su GitHub e successivamente, tramite il comando git clone, ho ottenuto una copia in locale su cui poter lavorare. Per ottimizzare il processo di build, in entrambi i progetti ho cancellato i file di test scritti dagli sviluppatori lasciando solamente i test da me implementati. Per quanto riguarda Fastjson, ho creato un nuovo progetto Maven importando la corretta dipendenza di Fastjson (versione 1.2.79).

In fondo al report sono state inserite le immagini delle osservazioni e dei risultati generati durante lo svolgimento del progetto e sono stati creati anche dei collegamenti a tali immagini. Ho deciso di inserire le immagini dei risultati per intero in modo da rendere il report più comprensibile.

FASTJSON.

Le classi di test che ho parametrizzato nel progetto di Fastjson sono state OrderedFieldTest.java e TypeUtilsTest.java. L'obiettivo è stato quello di prendere queste due classi di test scritte dagli sviluppatori di Fastjson e di trasformarle in classi con test parametrici. Iniziamo la discussione con la classe OrderedFieldTest.java che è stata rinominata nel mio progetto come TestOrderedFieldParameterized.java. La classe OrderedFieldTest.java contiene un solo metodo di test come mostrato nella [figura 1a](#). Nella nuova implementazione, all'interno del costruttore viene invocato il metodo **configure** che prende in input il valore atteso e tutti i parametri che dovranno essere passati ai metodi sotto test. In questo caso, si testano i metodi **parseObject** e **toJSONString** della classe JSON. Andando a vedere la documentazione, osservo che la classe JSON è astratta e che i due metodi testati sono entrambi statici. Di conseguenza, non devo istanziare la classe sotto test e posso invocare direttamente i metodi di JSON. Per questa classe, non ho parametrizzato l'assert ma ho utilizzato direttamente il valore atteso passato al configure.

La discussione per quanto riguarda la classe TypeUtilsTest.java risulta essere più complessa. La nuova implementazione della classe corrisponde all'interno del mio progetto alla classe TestTypeUtilsParameterized.java. La classe TypeUtilsTest.java contiene decine di metodi di test che si occupano di testare metodi di varie classi, tra cui JSON, TypeUtils e JSONObject. L'obiettivo che mi sono posto è stato quello di identificare le tipologie di test all'interno della classe e di parametrizzarle il più possibile. Ad esempio, nella [figura 2a](#) sono rappresentati due metodi della classe. In questo caso, viene testato da entrambi i metodi della classe il metodo "getObject" della classe JSONObject passandogli però parametri differenti. Tuttavia, questi due metodi possono rientrare all'interno di una stessa tipologia di test e di conseguenza potrei creare un test parametrico che inglobi entrambi i metodi. In realtà, all'interno della classe TypeUtilsTest.java ci sono numerosi metodi che rientrano all'interno di questa tipologia di test. Come si può vedere nella classe originale, ci sono molti metodi che possono essere raggruppati in un unico metodo attraverso la parametrizzazione. Ad esempio, i due metodi nella [figura 2a](#) possono essere parametrizzati come mostrato nella [figura 3a](#). Seguendo questo procedimento, sono riuscito a ridurre notevolmente il numero dei metodi di test andando a parametrizzare il più possibile. All'interno del costruttore della nuova classe, vengono chiamati i vari metodi di configure che hanno la responsabilità di creare l'istanza sotto test e di eseguire la configurazione dell'istanza stessa. Ad esempio, riprendendo sempre i due metodi della [figura 2a](#), il configure si occupa di allocare l'istanza di JSONObject e di configurarla invocando il metodo "put" con gli opportuni parametri. Come mostrato nella [figura 4a](#), ho deciso di utilizzare la libreria GSON per ricavarmi il valore atteso per alcune tipologie di test. In questo contesto, la libreria GSON viene considerata come un sistema ben funzionante il cui output può essere utilizzato per valutare il valore di ritorno del metodo sotto test. A questo punto, ho eseguito un'analisi sull'andamento della copertura strutturale in funzione del cambio dei valori per i parametri dei test. A tal proposito, ho creato delle nuove classi di test in cui ho utilizzato il runner Parametrized. Osservo che molti dei metodi sotto test si riconducono a chiamare il metodo "cast" della classe TypeUtils.java. Di conseguenza, ho deciso di studiare come varia la copertura per il metodo "cast" al variare del valore dei parametri. Inoltre, osservo che ci sono altri due metodi interessanti che non chiamano il metodo "cast", ossia castToBigInteger e castToBigDecimal che sono entrambi metodi della classe TypeUtils.java. Nella [figura 5a](#) è mostrata la coverage del metodo cast(Object, Class, ParserConfig) della classe TypeUtils che si ottiene con i casi di test implementati dagli sviluppatori di Fastjson. Dalla [figura 6a](#) osservo che ci sono numerose istruzioni di IF che non vengono prese dai casi di test implementati dagli sviluppatori di Fastjson. Ho deciso di progettare e implementare dei casi di test in modo da coprire tutti gli IF mostrati nella figura precedente. Eseguendo i nuovi casi di test, osservo nella [figura 7a](#) che le percentuali sono entrambe aumentate poiché sono riuscito ad entrare in tutti gli IF mostrati nella figura 6a. Noto che il valore della CONDITION COVERAGE è aumentato notevolmente perché ho coperto numerose condizioni con i nuovi casi di test. Nella [figura 8a](#) è mostrata la coverage per il metodo castToBigDecimal mentre nella [figura 9a](#) sono mostrate delle istruzioni IF che non

vengono coperte, entrambe queste informazioni sono relative ai casi di test implementati dagli sviluppatori nella classe TypeUtilsTest.java. Di conseguenza, ho progettato dei casi di test in cui passo come parametro null, un float, un double, un BigDecimal, una HashMap vuota e una stringa e in cui vado a parametrizzare l'assert. Con i nuovi casi di test, sono riuscito ad aumentare notevolmente le due percentuali come mostrato nella [figura 10a](#). Nella [figura 11a](#) è mostrata la coverage per il metodo castToBigInteger mentre nella [figura 12a](#) sono mostrate delle istruzioni IF che non vengono coperte dai casi di test implementati dagli sviluppatori nella classe TypeUtilsTest.java. Ho progettato dei casi di test in cui cerco di coprire tutti i rami IF che sono mostrati nella figura 12a. Eseguendo i nuovi casi di test sono riuscito ad aumentare notevolmente le percentuali come mostrato nella [figura 13a](#).

BOOKKEEPER.

Le classi individuate per il progetto Apache BookKeeper sono **BufferedChannel.java** e **BookieImpl.java**. I motivi relativi alla scelta delle classi sono stati i seguenti:

- Il numero degli sviluppatori che hanno partecipato all'implementazione delle classi è elevato. Ragionevolmente, è lecito pensare che la classe possa presentare con maggiore probabilità delle problematiche essendo solitamente gli stili dei programmatore diversi tra di loro.
- Le classi vengono modificate numerose volte durante le release del progetto.
- Le classi vengono modificate insieme ad un numero elevato di altri file.

La classe BufferedChannel.java fornisce un buffering layer per un FileChannel. Siccome la classe ha pochi metodi, ho deciso di testare più metodi della classe con lo scopo di raggiungere buone percentuali per quanto riguarda le metriche di adeguatezza che sono state scelte per il progetto. I due metodi della classe BufferedChannel.java su cui mi sono soffermato principalmente sono i seguenti:

- **public synchronized int read (ByteBuf dest, long pos, int length) throws IOException;**
- **public void write (ByteBuf src) throws IOException.**

La classe BookieImpl.java implementa un Bookie, ossia un BookKeeper server che gestisce frammenti di ledgers. Il metodo che ho deciso di testare è il seguente:

public static BookieSocketAddress getBookieAddress (ServerConfiguration conf)

Tale metodo ha il compito di restituire l'indirizzo configurato del Bookie server.

IDENTIFICAZIONE PARTIZIONI DI DOMINIO.

CLASSE: org.apache.bookkeeper.bookie.BookieImpl.java

METODO: **public static BookieSocketAddress getBookieAddress (ServerConfiguration conf)**

PARAMETRO	DESCRIZIONE	PARTIZIONI DI DOMINIO
conf	La configurazione gestisce le impostazioni lato server.	{null}, {"istanza valida"}, {"istanza non valida"}

Il parametro 'conf' è un tipo di dato complesso e di conseguenza bisogna identificare le partizioni applicando iterativamente i criteri generali ai dati nella struttura. I concetti di istanza valida e di istanza non valida sono fortemente legati al dominio applicativo in cui mi sto muovendo. Dalla documentazione si osserva che la classe ServerConfiguration ha numerosi campi di tipo stringa e alcuni campi di tipo ConfigKey. Ragionevolmente, attuando un approccio black-box ho supposto che molti di questi campi non vengano utilizzati dal metodo getBookieAddress in quanto sono relativi ad altre funzionalità del sistema BookKeeper. Ad esempio, non mi aspetto che al metodo getBookieAddress possa essere utile il campo FLUSH_INTERVAL. Basandomi sul nome del metodo e sulla documentazione, ho deciso di considerare i seguenti campi per il parametro conf:

- ADVERTISED_ADDRESS;
- BOOKIE_PORT;
- LISTENING_INTERFACE;
- USE_HOST_NAME_AS_BOOKIE_ID;
- USE_SHORT_HOST_NAME;
- ALLOW_LOOPBACK.

I campi che ho deciso di considerare sono tutti quanti delle stringhe. Per poter settare il valore di questi attributi, è necessario utilizzare il rispettivo metodo di setter. Andando sulla documentazione osservo che:

- Il metodo setAdvertisedAddress prende come parametro una stringa.
- Il metodo setBookiePort prende come parametro un intero che rappresenta la porta.
- Il metodo setListeningInterface prende come parametro una stringa che rappresenta una network interface.
- Il metodo setUseHostNameAsBookieID prende come parametro un boolean.

- Il metodo setUseShortHostName prende come parametro un boolean.
- Il metodo setAllowLoopback prende come parametro un boolean.

Seguendo le linee guida viste durante il corso, individuo le partizioni per ognuno dei parametri listati in precedenza basandomi sulle osservazioni fatte relative ai metodi di setter:

- Per il campo ADVERTISED_ADDRESS identifico le partizioni di dominio {"stringa valida"}, {"stringa non valida"}, {"null"}, {"stringa vuota"}.
- Per il campo BOOKIE_PORT identifico le partizioni di dominio { $0 \leq X < 65536$ }, { $X \geq 65536$ } e { $X < 0$ }.
- Per il campo LISTENING_INTERFACE identifico le partizioni di dominio {"stringa valida"}, {"stringa non valida"}, {"null"}, {"stringa vuota"}. In questo caso, per stringa valida si intende una stringa che rappresenta effettivamente una interfaccia di rete su cui il Bookie può mettersi in ascolto per le connessioni.
- Per i campi USE_HOST_NAME_AS_BOOKIE_ID, USE_SHORT_HOST_NAME e ALLOW_LOOPBACK considero le partizioni di dominio {true} e {false}.

CLASSE: org.apache.bookkeeper.bookie.BufferedChannel.java

METODO: public synchronized int read (ByteBuf dest, long pos, int length)

PARAMETRO	DESCRIZIONE	PARTIZIONI DI DOMINIO
dest	Buffer di destinazione	{buffer vuoto}, {buffer non vuoto}, {null}
pos	Posizione da cui iniziare la lettura	{ <0 }, { ≥ 0 }
length	Numero di byte che devono essere letti	{ <0 }, { ≥ 0 }

Un oggetto ByteBuf astrae una sequenza di zero o più byte e di conseguenza ho individuato le tre partizioni {buffer vuoto}, {buffer non vuoto} e {null}. Il parametro 'pos' rappresenta la posizione da cui partire per leggere i dati. È ragionevole individuare le due classi di equivalenza { <0 } e { ≥ 0 } aspettandomi una logica di errore nel momento in cui il valore del parametro è negativo. La posizione a partire dalla quale deve avvenire la lettura dovrebbe essere maggiore o uguale a zero. Per quanto riguarda il parametro 'length', non ci sono informazioni sulla documentazione del progetto. Ragionevolmente, ho supposto che il parametro 'length' rappresenti il numero di byte che devono essere letti. Ovviamente, questo valore deve essere positivo oppure zero nel caso in cui non si voglia leggere alcun byte.

CLASSE: org.apache.bookkeeper.bookie.BufferedChannel.java

METODO: public void write (ByteBuf src)

PARAMETRO	DESCRIZIONE	PARTIZIONI DI DOMINIO
src	Buffer contenente i dati che dovranno essere scritti	{buffer vuoto}, {buffer non vuoto}, {null}

Le motivazioni per la scelta delle partizioni di dominio per il parametro dest del metodo read sono valide anche per il partizionamento del dominio del parametro src.

BOUNDARY ANALYSIS.

Nel progetto è stata adottata la tecnica boundary-value per la scelta di uno o più valori rappresentativi per una classe di equivalenza.

CLASSE: org.apache.bookkeeper.bookie.BookieImpl.java

METODO: public static BookieSocketAddress getBookieAddress (ServerConfiguration conf)

PARAMETRO	BOUNDARY-VALUES
ADVERTISED_ADDRESS	'192.168.1.40', '', 'it is not an address', null
BOOKIE_PORT	65536, 65535, 5000, 1025, 0, -1, -1000
LISTENING_INTERFACE	"eth2", "eth1", "eth0", "it is not a network interface", null
USE_HOST_NAME_AS_BOOKIE_ID	true, false
USE_SHORT_HOST_NAME	true, false
ALLOW_LOOPBACK	true, false

CLASSE: org.apache.bookkeeper.bookie.BufferedChannel.java

METODO: public synchronized int read (ByteBuf dest, long pos, int length)

PARAMETRO	BOUNDARY-VALUES
dest	{buffer vuoto}, {buffer con piccola capacità}, {buffer con capacità media}, {null}

pos	-567896, -1, 0, 500
length	-5000, -1, 0, 128, 256, 1024

CLASSE: org.apache.bookkeeper.bookie.BufferedChannel.java

METODO: public void write (ByteBuf src)

PARAMETRO	BOUNDARY-VALUES
src	{buffer vuoto}, {buffer con piccola capacità}, {buffer con capacità media}, {null}

IDENTIFICAZIONE CASI DI TEST.

CLASSE: org.apache.bookkeeper.bookie.BufferedChannel.java

METODO: public void write (ByteBuf src)

METODO: public synchronized int read (ByteBuf dest, long pos, int length)

Per quanto riguarda il metodo read, è stata necessaria una fase di configurazione in cui ho scritto all'interno del canale un certo numero di byte in modo da poterli leggere successivamente testando così il metodo di lettura. Come primo step, ho creato un file temporaneo che verrà distrutto al termine della macchina virtuale. Successivamente, su questo file temporaneo ho creato un canale per poter eseguire operazioni di scrittura e di lettura. A questo punto ho istanziato la classe sotto test, BufferedChannel, e scritto dei byte random all'interno del canale senza eseguire l'operazione di flush sul file. Dalla [documentazione](#), osservo che è possibile stabilire la capacità del buffer di scrittura in fase di creazione dell'istanza, specificandola come parametro del costruttore. Il campo writeBuffer della classe BufferedChannel.java rappresenta il buffer utilizzato per le operazioni di scrittura mentre il campo writeCapacity rappresenta la capacità del write buffer. Noto che ai fini del testing, ovviamente c'è un forte legame tra il numero di byte che vado a scrivere e il numero di byte che successivamente andrò a leggere. Dalla documentazione, osservo che il campo writeBuffer è di tipo io.netty.buffer.ByteBuf. Un oggetto ByteBuf ha due variabili puntatore per supportare le operazioni di lettura e di scrittura sequenziali. Queste due variabili sono readerIndex per le operazioni di lettura e writerIndex per le operazioni di scrittura. La relazione che c'è tra queste variabili è la seguente: $0 \leq \text{readerIndex} \leq \text{writerIndex} \leq \text{capacity}$. Di conseguenza, se eseguo una scrittura di N byte, il valore di writerIndex sarà pari ad N e in fase di lettura non potrò leggere più di N byte. Nella progettazione dei casi di test, ho deciso di riportare anche ulteriori parametri, oltre a quelli richiesti dai metodi sotto test. Questi ulteriori parametri riguardano la dimensione del buffer che utilizzo per la scrittura, la capacità minima del write buffer passata in input al momento della creazione dell'istanza e un parametro che mi permette di specificare il bound per il numero massimo di byte che non sono stati ancora mandati in persistenza, attivando così operazioni di flush regolari. Basandomi su queste osservazioni, per il metodo "read" ho deciso di progettare i seguenti casi di test:

1. {capacity = 2048, ubb = 0, numberByteWritten = 256, sizeReadBuf = 256, pos = 0, length = 257}: il writer buffer ha una capacità minima pari a 2048 byte. Con il valore ubb = 0, impedisco l'esecuzione dell'operazione di flush sul file. Vengono scritti 256 byte e la dimensione del ByteBuf che passo al metodo read è esattamente pari a 256 byte. Mi aspetto una eccezione in quanto richiedo la lettura di 257 byte a partire dalla posizione zero quando in realtà ne sono stati scritti 256.
2. {2048, 0, 256, 1024, 0, 1024}: mi aspetto una eccezione in quanto tento di leggere 1024 byte quando ne sono stati scritti solamente 256. A differenza del caso di test precedente, la dimensione del buffer per la lettura è strettamente maggiore del numero di byte che sono stati scritti.
3. {2048, 0, 256, 64, 0, 256}: in questo caso di test, sono stati scritti 256 byte e tento di leggere tutti i 256 byte scritti partendo dalla posizione zero. Tuttavia, mi aspetto una eccezione in quanto la dimensione del buffer di lettura (64 byte) è strettamente inferiore rispetto al numero di byte che voglio leggere.
4. {2048, 0, 256, 0, 0, 256}: in questo caso di test, la dimensione del buffer in cui andranno inseriti i byte letti è pari a zero (buffer vuoto). Mi aspetto che il metodo lanci una eccezione.
5. {1024, 0, 256, 256, 0, 256}: mi aspetto che il valore di ritorno del metodo read sia pari a 256. Sono stati scritti 256 byte e successivamente vengono letti tutti i 256 byte scritti in precedenza partendo dalla posizione zero. Il buffer utilizzato per la lettura è in grado di contenere i byte che dovranno essere letti.
6. {2048, 0, 256, 0, 0, 0}: in questo caso di test, sono stati scritti 256 byte e si richiede la lettura di zero bytes. Anche se la dimensione del buffer utilizzato per la lettura è pari a zero, mi aspetto che il metodo non lanci una eccezione e mi ritorni il valore zero senza aver letto alcun byte.

7. {2048, 0, 256, 512, 0, 256}: mi aspetto che il valore di ritorno del metodo read sia pari a 256. Non si devono riscontrare problemi se la dimensione del buffer utilizzato per la lettura (512 byte) è strettamente maggiore del numero dei byte che dovranno essere letti.

Per quanto riguarda il metodo write, sono stati considerati i seguenti casi di test:

8. Il parametro in input è null. Indipendentemente dalla capacità minima del write buffer e dall'esecuzione della operazione di flush su file, mi aspetto che il metodo lanci una eccezione. I casi di test progettati sono {capacity=2048, ubb=0} e {1024, 1}.
9. Il parametro di input è un buffer vuoto. Mi aspetto che il metodo termini immediatamente senza lanciare alcuna eccezione. Siccome il buffer è vuoto, non verrà scritto alcun byte. I casi di test progettati sono {capacity=512, ubb=0, numberByteToWrite=0}.
10. Il parametro di input è un buffer che ha una dimensione strettamente maggiore di zero e minore della capacità minima del write buffer. I casi di test progettati sono {capacity=10000, ubb=0, numberByteToWrite=8192} e {2048, 0, 512}.

Per verificare la correttezza del metodo write, ho utilizzato le eccezioni e ho verificato attraverso un'operazione di lettura se è stato scritto il numero corretto di byte. Le metriche scelte per il progetto BookKeeper sono la STATEMENT COVERAGE e la CONDITION COVERAGE, entrambe fornite da JaCoCo. Implementando i casi di test progettati finora ed eseguendoli, ottengo dal report di JaCoCo i seguenti valori per le due metriche:



Per quanto riguarda PIT, il report è mostrato nella [figura a](#). Dai risultati del report di PIT, osservo che sono presenti numerosi NO_COVERAGE. Nella seconda iterazione, cercherò di progettare dei casi di test che coprano le porzioni di codice su cui sono state generate le mutazioni e che non sono state raggiunte dalla mia test suite. Ad esempio, come mostrato nella [figura c](#), i mutanti non sono stati identificati in quanto i casi di test progettati ed implementati finora non sono stati in grado di coprire questa parte di codice. Il report generato da Badua è mostrato nella [figura b](#). Osservo che per il metodo write sono riuscito a coprire esattamente la metà delle coppie def-use. Mentre per il metodo read, il numero delle coppie che sono state coperte è inferiore e di conseguenza nella seconda iterazione dovrò cercare di progettare dei casi di test che mi aumentino la percentuale di coppie coperte per il metodo di lettura.

Nella seconda iterazione, ho progettato i seguenti casi di test per il metodo read:

1. {2048, 0, 256, 256, -567896, 128}: mi aspetto che il metodo read lanci una eccezione in quanto sto passando come valore del parametro pos un numero negativo.
2. {2048, 0, 256, 256, -1, 128}: mi aspetto che il metodo read lanci una eccezione in quanto sto passando come valore del parametro pos un numero negativo.
3. {2048, 0, 256, 256, 0, -1}: mi aspetto che il valore di ritorno del metodo read sia pari a zero. Il valore del parametro length è pari a -1 e di conseguenza non deve essere letto alcun byte.
4. {2048, 0, 256, 256, 0, -5000}: mi aspetto che il valore di ritorno del metodo read sia pari a zero. Il valore del parametro length è pari a -5000 e di conseguenza non deve essere letto alcun byte. A differenza del caso di test precedente, ho utilizzato un valore molto più piccolo per il parametro length.
5. {255, 0, 256, 256, 0, 256}: in questo caso di test, sto passando come valore minimo della capacità del buffer di scrittura 255 byte che è strettamente inferiore del numero di byte che vado a scrivere (256 byte). Mi aspetto che il metodo read ritorni come valore 256.
6. {1024, 0, 512, 700, 500, 6}: in questo caso di test, scrivo 512 byte e tento di leggere 6 byte partendo dalla posizione 500. Siccome dalla posizione 500 effettivamente sono disponibili 6 byte per la lettura, il metodo read mi deve restituire il valore 6. In questo caso, il buffer utilizzato per la lettura ha la capacità sufficiente per memorizzare i byte che devono essere letti.

Inoltre, ho progettato dei casi di test per il metodo write:

7. A seguito della scrittura, viene eseguita l'operazione di flush dei dati sul file e si tenta di chiudere più volte il canale. Mi aspetto che i byte vengano scritti su file e che il canale si chiuda correttamente senza lanciare eccezioni.
8. {capacity=256, ubb=1, numberByteToWrite=128}: mi aspetto che vengano scritti 128 bytes e che venga eseguita l'operazione di flush in quanto il valore del parametro ubb che rappresenta il bound del numero di byte che non sono stati ancora mandati in persistenza è strettamente maggiore di zero e minore di 128 che sono i bytes che vengono scritti.
9. {capacity=256, ubb=0, numberByteToWrite=512}: in questo caso di test la capacità minima del write buffer (256) è strettamente inferiore rispetto ai byte che vado a scrivere. Mi aspetto che vengano scritti 512 byte e che vengano eseguite delle operazioni di flush per liberare il writeBuffer durante la scrittura.

Per i casi di test in cui vado a eseguire il flush sul file, vengono fatti i seguenti controlli:

- a. Viene controllato il numero di byte all'interno del write buffer prima dell'esecuzione dell'operazione di flush.
- b. Viene controllato il numero di byte all'interno del write buffer dopo l'esecuzione dell'operazione di flush.
- c. Viene controllato il numero di byte scritti all'interno del file dopo l'esecuzione dell'operazione di flush.
- d. Viene controllato il valore del campo position della classe BufferedChannel successivamente al flush dei dati su file.

Nell'esecuzione dei casi di test della seconda iterazione, osservo che il caso di test n°6 fallisce. Osservo che il valore di ritorno del metodo read è pari a 12 anziché 6. Ragionevolmente, ho pensato che il valore 12 corrispondesse a $512 - 500$, ossia al numero di byte che sono stati scritti a partire dalla posizione 500, passata in input al metodo. Dalla documentazione, non trovo informazioni relative al parametro length del metodo read. Dal nome del parametro 'length', ragionevolmente ho ipotizzato che il suo valore corrisponda al numero di byte che dovranno essere letti a partire dalla posizione rappresentata dal parametro 'pos'. Siccome l'approccio black-box non è sufficiente per risolvere questo problema, sono andato a vedere direttamente il codice del metodo read. Noto che vengono eseguiti solamente un controllo all'interno di un ciclo sul valore del parametro length per verificare se è strettamente maggiore di zero e un decremento del valore di length pari al numero di byte che sono stati copiati nell'iterazione. Tuttavia, il numero di byte copiati non viene calcolato utilizzando il parametro length ma considerando il minimo tra il numero di byte ancora scrivibili all'interno del buffer utilizzato per la lettura (quello passato in input al metodo read) e il numero di byte all'interno del write buffer a partire dalla posizione specificata fino al valore del writerIndex del write buffer stesso. Probabilmente, sarebbe stato opportuno specificare il significato del parametro length in modo da evitare incomprensioni per chi andrà ad utilizzare il metodo. Inoltre, osservo che i casi di test eseguiti nella prima iterazione sono corretti in quanto effettivamente il valore del parametro length è pari al numero di byte che vengono scritti e la lettura parte dalla posizione zero.

Eseguendo i casi di test implementati nella seconda iterazione, ottengo i seguenti risultati dal report di JaCoCo:



Ampliando la test suite, sono riuscito a raddoppiare il numero delle istruzioni coperte per la classe e sono riuscito ad aumentare il valore della CONDITION COVERAGE. Il report di Badua relativo alla seconda iterazione è mostrato nella [figura d](#). Come si può vedere dal report, il numero di coppie def-use che non sono state coperte nel metodo write si è ridotto solamente a sei. Il caso di test {capacity=256, ubb=1, numberByteToWrite =128} mi ha permesso di coprire numerose coppie def-use. Ad esempio, mi ha permesso di entrare all'interno del costrutto if che si trova a riga 135 (ho abilitato delle operazioni di flush regolari poiché ubb=1>0) e quindi di coprire la coppia def-use mancante per la variabile "this.doRegularFlushes". Mi ha permesso di entrare all'interno del costrutto if a riga 143 coprendo la coppia def-use per "shouldForceWrite" e infine di entrare all'interno del costrutto if a riga 137 in quanto il numero di byte che sono stati scritti e che ancora non sono stati mandati in persistenza è strettamente maggiore di 1. Il caso di test {capacity=256, ubb=0, numberByteToWrite=512}, mi ha permesso di entrare all'interno del costrutto if a riga 130 coprendo così l'undicesima coppia def-use e di eseguire due volte l'iterazione while andando così a coprire la terza coppia def-use partendo dall'ultima. Inoltre, con l'introduzione dei casi di test per il metodo read, sono riuscito a diminuire notevolmente il numero delle coppie non coperte. Ad esempio, prendiamo il caso di test n°5. Siccome vengono scritti 256 byte e la dimensione minima del writeBuffer è strettamente inferiore (255 byte), viene eseguita l'operazione di flush nella fase di configurazione e viene aggiornata la posizione iniziale per la scrittura all'interno del write Buffer. Di conseguenza, nel metodo read riesco ad entrare all'interno del ramo "else" e successivamente nel ramo "else if" a riga 265 andando così a coprire numerose coppie def-use come mostrato nel report. Il report di PIT relativo alla seconda iterazione è mostrato nella [figura e](#). Come si può osservare dal report, con l'ampliamento della test suite il numero dei mutanti KILLED è aumentato. Inoltre, grazie ai nuovi casi di test sono riuscito ad aumentare il numero dei mutanti KILLED per la porzione di codice mostrato nella [figura c](#) raggiungendo uno degli obiettivi che mi sono posto nella iterazione precedente.

Nella terza iterazione, ho cercato di risolvere l'incomprensione che è nata a causa di una scarsa documentazione. Seguendo un approccio white box, sono riuscito ad individuare i valori di ritorno corretti per quanto riguarda il metodo read dando un significato diverso al parametro length. Supponendo che il valore del parametro pos sia maggiore o uguale a zero e che il valore del parametro length sia strettamente maggiore di zero (altrimenti si ricade nei casi già affrontati nelle due iterazioni precedenti) ho ricavato il comportamento del metodo read come segue:

caso 1: la differenza tra il numero di byte scritti e il valore del parametro 'pos' è minore o uguale al numero di byte scrivibili nel buffer di lettura e la differenza è maggiore o uguale al valore del parametro length. In questo

caso, la dimensione del buffer di lettura ha una capacità sufficiente per poter eseguire l'operazione di lettura e i byte che vengono letti sono quelli dal 'pos' fino all'ultimo byte scritto.

Caso 2: la differenza tra il numero di byte scritti e il valore del parametro 'pos' è strettamente maggiore del numero di byte scrivibili nel buffer di lettura ma il numero di byte scrivibili nel buffer di lettura è maggiore o uguale al valore del parametro length. In questo caso il numero dei byte letti è pari al numero di byte scrivibili all'interno del buffer di lettura.

Caso 3: la differenza tra il numero di byte scritti e il valore del parametro 'pos' è minore o uguale al numero di byte scrivibili nel buffer di lettura ma è anche strettamente minore di length. In questo caso mi aspetto che il metodo lanci una eccezione in quanto sto tentando di leggere più di quanto è disponibile.

Caso 4: la differenza tra il numero di byte scritti e il valore del parametro 'pos' è strettamente maggiore del numero di byte scrivibili nel buffer di lettura e il numero di byte scrivibili nel buffer di lettura è strettamente minore di length. Mi aspetto che il metodo lanci una eccezione in quanto sto cercando di leggere più byte di quanti ne posso scrivere all'interno del buffer utilizzato per la lettura.

L'implementazione relativa al calcolo del valore atteso è mostrata nella [figura f](#). I casi di test progettati nelle iterazioni precedenti, che rientrano in uno di questi quattro casi individuati, risultano essere corretti. Implementando i nuovi casi di test non sono riuscito ad aumentare le percentuali con JaCoCo, con Badua e con PIT.

CLASSE: org.apache.bookkeeper.bookie.BookieImpl.java

METODO: `public static BookieSocketAddress getBookieAddress (ServerConfiguration conf)`

Per tutti i casi di test è stata necessaria l'esecuzione di una fase di configurazione prima di poter invocare il metodo `getBookieAddress`. Questo perché ho dovuto settare il valore degli attributi del parametro `conf` prima di passarlo al metodo sotto test. Inoltre, osservo che il metodo `getBookieAddress` è statico e di conseguenza non devo istanziare la classe `BookieImpl` per poterlo invocare. I casi di test progettati nelle varie iterazioni si differenziano tra di loro in base a come è stato settato lo stato del parametro `conf`. Come già detto nella fase di "IDENTIFICAZIONE PARTIZIONI DI DOMINIO", è stato considerato un sottoinsieme di tutti gli attributi della classe `ServerConfiguration` che ragionevolmente sono coinvolti nel metodo `getBookieAddress`. Per capire il comportamento del metodo sotto test, ho sia utilizzato il suo valore di ritorno sia l'eccezione di tipo `UnknownHostException` che viene lanciata dal metodo stesso.

Nella prima iterazione ho deciso di progettare dei casi di test che portino il metodo a fallire considerando le seguenti configurazioni del parametro `conf`:

1. `{ADVERTISED_ADDRESS="" ,BOOKIE_PORT=-1,LISTENING_INTERFACE="eth0" ,USE_HOST_NAME_AS_BOOKIE_ID=false,USE_SHORT_HOST_NAME=false,ALLOW_LOOPBACK = false}`: Data la seguente configurazione, mi aspetto che il metodo lanci una eccezione in quanto il numero di porta non è valido poiché negativo.
2. `{null, 1025, "it is not a network interface", false, false, false}`: Data la seguente configurazione, mi aspetto che il metodo lanci una eccezione in quanto l'interfaccia di rete non è valida.
3. `{"", -1000, null, false, false, false}`: Data la seguente configurazione, mi aspetto che il metodo lanci una eccezione in quanto il numero di porta non è valido.
4. `{"", 65536, "eth1", false, false, false}`: Data la seguente configurazione, mi aspetto che il metodo lanci una eccezione in quanto il numero di porta è al di fuori del range dei valori validi.

A questo punto, sono passato a progettare dei casi di test in cui vado a testare le varie interfacce di rete della macchina su cui sto eseguendo i test e vado a testare diversi numeri di porta:

5. `{"", 1025, "eth0", false, false, false}`: Mi aspetto che il metodo non lanci alcuna eccezione e che mi ritorni un `BookieSocketAddress` con numero di porta pari a 1025. La macchina su cui si sta eseguendo il test dispone dell'interfaccia di rete specificata.
6. `{"", 5000, "eth1", false, false, false}`: Mi aspetto che il metodo non lanci alcuna eccezione e che mi ritorni un `BookieSocketAddress` con numero di porta pari a 5000. La macchina su cui si sta eseguendo il test dispone dell'interfaccia di rete specificata.
7. `{"", 65535, "eth2", false, false, false}`: Mi aspetto che il metodo non lanci alcuna eccezione e che mi ritorni un `BookieSocketAddress` con numero di porta pari a 65535. La macchina su cui si sta eseguendo il test dispone dell'interfaccia di rete specificata.

Eseguendo i casi di test progettati in questa iterazione ottengo i seguenti risultati dal report di JaCoCo:



Le metriche scelte sono la STATEMENT COVERAGE e la CONDITION COVERAGE. Come si può vedere dal report, sono riuscito a raggiungere circa il 50% per entrambe le metriche. Nel caso di Badua, non sono riuscito a risolvere un errore nella generazione del report e di conseguenza non l'ho potuto utilizzare per quanto riguarda il metodo `getBookieAddress`. Purtroppo, non sono riuscito a trovare informazioni relative al mio problema sulla documentazione o su altre fonti legate al prodotto. Il report di PIT è mostrato nella [figura g](#). Siccome la classe `BookieImpl.java` ha numerosi metodi, ho deciso di forzare PIT a creare le mutazioni solamente sul metodo sotto test in modo da ridurre il tempo di esecuzione. Per fare ciò ho utilizzato i tag **"excludedMethods"** e **"excludedMethod"** in cui ho inserito il nome dei metodi che dovranno essere esclusi. Come si può vedere dal report, non sono riuscito ad avere un alto valore della mutation coverage. Un motivo è il fatto che non sono riuscito a coprire con la test suite progettata un buon numero di linee di codice, come si può osservare anche dal report di JaCoCo. Di conseguenza, il mio obiettivo sarà quello di migliorare la STATEMENT COVERAGE e la CONDITION COVERAGE cercando anche di aumentare il numero dei mutanti KILLED. Nel caso di PIT, ho implementato la funzionalità sperimentale detta **analisi incrementale**. In questo modo, PIT tiene traccia di determinate informazioni per evitare di ripetere l'analisi quando i risultati possono essere dedotti logicamente. Per utilizzare l'analisi incrementale, ho seguito la [documentazione](#) e alcuni siti collegati al prodotto andando a inserire all'interno del profilo da me creato i tag `historyInputFile` e `historyOutputFile`. Attraverso questi due parametri vengono indicate le posizioni da cui leggere e scrivere i risultati dell'analisi delle mutazioni. Nel mio caso, ho deciso di utilizzare la stessa posizione per entrambe.

Nella seconda iterazione, ho progettato i seguenti casi di test che determinano una corretta esecuzione del metodo:

1. `{"192.168.1.40", 1025, "eth0", false, false, false}`: Mi aspetto che il metodo ritorni un `BookieSocketAddress` con numero di porta pari a 1025 e un hostname uguale a "192.168.1.40".
2. `{"192.168.1.40", 65535, "it is not a network interface", false, false, false}`: Mi aspetto che il metodo non lanci una eccezione in quanto ho un valore valido per Advertised Address, indipendentemente dalla interfaccia di rete. Mi aspetto che il metodo ritorni un `BookieSocketAddress` con numero di porta pari a 65535 e hostname uguale a "192.168.1.40".
3. `{"", 65535, "eth2", true, false, false}`: Nel caso della macchina su cui sto eseguendo i test, il valore dell'hostname è pari a "MSI". Poiché il parametro `USE_HOST_NAME_AS_BOOKIE_ID` ha valore uguale a `true`, mi aspetto un `BookieSocketAddress` con numero di porta pari a 65535 e hostname uguale a "MSI".
4. `{"", 5000, "eth1", true, true, false}`: Siccome i parametri `USE_HOST_NAME_AS_BOOKIE_ID` e `USE_SHORT_HOST_NAME` sono pari a `true`, allora viene utilizzato lo short hostname che è uguale all'hostname tagliato al primo punto "." all'interno della stringa. Nel mio caso, l'hostname è uguale a "MSI" che al suo interno non ha alcun punto. Quindi lo short hostname sarà uguale a "MSI" e mi aspetto che il metodo ritorni un `BookieSocketAddress` con numero di porta pari a 5000 e hostname uguale a "MSI".

Inoltre, ho progettato il seguente caso di test che porta il metodo a lanciare una eccezione:

5. `{"192.168.1.40", -1, "eth0", false, false, false}`: Mi aspetto che il metodo lanci una eccezione in quanto il valore della porta non è valido.

Eseguendo i casi di test progettati nella seconda iterazione, sono riuscito ad aumentare il valore delle metriche come mostrato nel seguente report di JaCoCo:



Il report di PIT è mostrato nella [figura h](#). Ampliando la test suite, sono riuscito a coprire un maggior numero di righe di codice e ad aumentare di poco la mutation coverage. Per risolvere il problema dei `TIMED_OUT` ho aggiunto all'interno del profilo relativo a PIT il tag **timeoutConstant** specificando un valore maggiore rispetto a quello di default che era stato utilizzato in precedenza.

Tuttavia, per quanto riguarda questo metodo ho riscontrato un problema che non sono riuscito a risolvere. Per avere successo nella fase di build con le Github Actions, sono stato costretto a disattivare i test relativi al metodo `getBookieAddress` in quanto fanno uso di un indirizzo IP, di interfacce di rete e di hostname che sono dipendenti dalla macchina su cui si stanno eseguendo i test e che quindi mi portano la build al fallimento. Non sono riuscito a trovare un modo per rendere i test indipendenti dalla macchina host. Per questo metodo, tutte le tecniche sono state sperimentate in locale.

Per il progetto Apache BookKeeper, ho deciso di implementare una ulteriore classe che prevede l'utilizzo di Mockito. La classe di test che utilizza la libreria si chiama TestBufferedMock.java. Il metodo "testWriteReadableBytesMock" utilizza Mockito per mockare la classe ByteBuf e per verificare se sull'istanza di ByteBuf passata al metodo write della classe BufferedChannel.java viene invocato il metodo readableBytes. Il metodo "testRead" mi consente di testare il metodo "read" della classe BufferedChannel.java. Ho utilizzato Mockito per mockare la classe ByteBuf e ho creato una nuova Answer per il metodo "writeBytes". Vado a verificare se vengono letti tutti quanti i bytes che sono stati scritti in precedenza all'interno del canale. Per questi test, ho utilizzato tutte le informazioni che sono state ricavate nelle iterazioni per la classe BufferedChannel.java.

ZOOKEEPER.

Le classi individuate per il progetto Apache ZooKeeper sono **PathUtils.java** e **DataTree.java**.

Analizzando il file Excel prodotto nel progetto del Prof. Falessi, ho notato che la classe DataTree.java è risultata essere difettosa in numerose release. Inoltre, è una classe che ha un ruolo chiave all'interno del sistema in quanto mantiene due importanti strutture dati parallele: una tabella hash che mappa i percorsi completi dei nodi ai DataNode e un albero di DataNode. I metodi che ho scelto di testare relativi a questa classe sono:

- `public void createNode (final String path, byte [] data, List<ACL> acl, long ephemeralOwner, int parentCVersion, long zxid, long time);`
- `public void deleteNode (String path, long zxid);`

Osservo che i due metodi hanno come tipo di ritorno void. Per capire se effettivamente l'esecuzione di questi metodi ha avuto l'effetto atteso, potrei analizzare lo stato delle strutture dati che sono coinvolte nell'operazione. Ad esempio, per il metodo che crea il nodo mi aspetto che la dimensione della struttura dati aumenti mentre per il metodo che elimina il nodo mi aspetto che la dimensione della struttura dati diminuisca. Un altro modo per verificare se il metodo ha il comportamento che mi aspetto, è quello di utilizzare le eccezioni che il metodo stesso può lanciare. Osservo che entrambi i metodi sono stati sviluppati da un gran numero di programmatori. La classe PathUtils.java ha un ruolo fondamentale all'interno del sistema in quanto verifica la validità della stringa che rappresenta il path di uno ZNode. Il metodo di questa classe che ho deciso di testare è il seguente:

`public static void validatePath (String path) throws IllegalArgumentException`

La responsabilità di questo metodo è quella di validare il path di uno ZNode passato in input come stringa. Osservo che il metodo ha void come tipo di ritorno. Tuttavia, dalla segnatura del metodo e dalla documentazione osservo che viene lanciata una specifica eccezione nel momento in cui riceve un path per un nodo in ZooKeeper che non è valido. Di conseguenza, posso approfittare delle eccezioni lanciate dal metodo per capire se il metodo sotto test si comporta nel modo atteso.

IDENTIFICAZIONE PARTIZIONI DI DOMINIO.

CLASSE: org.apache.zookeeper.common.PathUtils.java

METODO: `public static void validatePath (String path)`

PARAMETRO	DESCRIZIONE	PARTIZIONI DI DOMINIO
path	Path relativo ad uno ZNode	{"null"}, {"stringa valida"}, {"stringa non valida"}, {"stringa vuota"}

Seguendo le linee generali viste durante il corso, ho individuato le partizioni rappresentate all'interno della tabella precedente. Osservo che i concetti di stringa valida e di stringa non valida sono strettamente legati al dominio applicativo in cui mi sto muovendo. ZooKeeper ha un namespace gerarchico e ogni nodo può avere dati associati ad esso e ai propri figli. Tuttavia, non è possibile utilizzare qualsiasi carattere Unicode per costruire il path che identifica uno ZNode. Per la definizione delle stringhe valide e non valide sono andato a cercare informazioni nella documentazione del progetto. Il namespace fornito da ZooKeeper è molto simile a quello di un file system standard. Il nome di uno ZNode è una sequenza di elementi di percorsi separati da uno slash (/). Ogni nodo all'interno del namespace di ZooKeeper è identificato da un path. Il path deve iniziare con il carattere slash (/), che rappresenta la radice dell'albero (si veda la [figura 1](#)).

CLASSE: org.apache.zookeeper.server.DataTree.java

METODO: `public void createNode (final String path, byte [] data, List<ACL> acl, long ephemeralOwner, int parentCVersion, long zxid, long time);`

PARAMETRO	DESCRIZIONE	PARTIZIONI DI DOMINIO
path	Path relativo ad uno ZNode	{"null"}, {"stringa valida"}, {"stringa non valida"}, {"stringa vuota"}
data	Dati associati al nodo	{"array non vuoto"}, {"array vuoto"}, {"null"}
acl	Lista di Access Control List	{"lista vuota"}, {"lista non vuota"}, {"null"}
ephemeralOwner	Specifica la natura dello ZNode	{0}, {>0}, {0xFF0000XXXXXXXXXXXX}, {0x8000000000000000}, R - {0xFF0000XXXXXXXXXXXX} U {0x8000000000000000}
parentCVersion	-	{<0}, {≥0}
zxid	Identificativo transazione ZooKeeper	{<0}, {≥0}
time	valore del timestamp	{<0}, {≥0}

Le motivazioni sul partizionamento del dominio del parametro **path** presentate per il metodo validatePath sono valide anche per il metodo createNode. Il parametro **data** è un array di byte. Dalla documentazione, non ho trovato alcun vincolo sul contenuto dell'array di dati che viene memorizzato dal nodo. Di conseguenza, non potendo determinare quando un array di dati risulta essere non valido, ho deciso di identificare le due partizioni {"array non vuoto"} e {"array vuoto"}. Il valore del parametro **ephemeralOwner** dipende dalla natura dello ZNode. Originariamente, con questo parametro era possibile specificare se lo ZNode fosse effimero e quindi quale sessione lo avesse creato. Attraverso una proprietà di sistema, è possibile abilitare ulteriori funzionalità come i nodi TTL. È possibile distinguere le seguenti tipologie di nodi: container, TTL, effimeri standard e non effimeri. Se il parametro assume valore pari a zero, allora il nodo non è effimero. Se la proprietà di sistema viene attivata e si fornisce un valore del parametro che rispetta il formato 0xFF0000XXXXXXXXXX, allora si sta creando un nodo TTL. I valori rappresentati con X servono a stabilire il TTL espresso in millisecondi. Per creare un nodo container, il valore del parametro deve essere pari a 0x8000000000000000L e non si ha l'obbligo di abilitare la proprietà di sistema richiesta per la creazione dei nodi TTL. Infine, per creare un tradizionale nodo effimero, devo passare come valore del parametro l'ID della sessione. Il parametro **zxid** rappresenta l'identificativo della transazione in ZooKeeper. È composto da due gruppi di 32 bit di cui il primo rappresenta l'epoca ed il secondo rappresenta un contatore. Siccome è un identificativo, ragionevolmente ho supposto che lo zero possa essere un valore valido e di conseguenza ho identificato le due partizioni {<0}, {≥0}. In ZooKeeper, ogni nodo dispone di ACL che limitano chi può fare cosa su tale nodo. ZooKeeper utilizza le ACL per controllare l'accesso ai suoi ZNodes. ZooKeeper supporta i seguenti permessi:

- CREATE: è possibile creare un nodo figlio.
- READ: è possibile ottenere i dati da un nodo e listare i suoi figli.
- WRITE: è possibile settare i dati per un nodo.
- DELETE: è possibile eliminare un nodo figlio.
- ADMIN: è possibile impostare i permessi.

Inoltre, sono definite delle ACL, che il sistema mette a disposizione, per cui sono stabiliti specifici permessi: OPEN_ACL_UNSAFE, CREATOR_ALL_ACL e READ_ACL_UNSAFE. Basandomi su queste informazioni ho deciso di identificare le partizioni descritte nella tabella relative al parametro **acl**, considerando come liste vuote quelle messe a disposizione direttamente dal sistema. Il CVersion rappresenta il numero di modifiche che si sono avute sui figli di uno ZNode. Di conseguenza, rappresentando il numero delle modifiche, per il parametro **parentCVersion** ho deciso di inserire il valore zero all'interno della partizione {>=0} in quanto è possibile che i figli di uno ZNode non siano soggetti a modifiche.

CLASSE: org.apache.zookeeper.server.DataTree.java

METODO: public void deleteNode (String path, long zxid);

PARAMETRO	DESCRIZIONE	PARTIZIONI DI DOMINIO
path	Path relativo ad uno ZNode	{"null"}, {"stringa valida"}, {"stringa non valida"}, {"stringa vuota"}
zxid	Identificativo transazione ZooKeeper	{<0}, {≥0}

Per i due parametri del metodo deleteNode valgono gli stessi ragionamenti fatti nel metodo createNode per il partizionamento del dominio.

BOUNDARY ANALYSIS.

CLASSE: org.apache.zookeeper.common.PathUtils.java

METODO: public static void validatePath (String path)

PARAMETRO	BOUNDARY-VALUES
path	null, "/znode1", "/a/b/c", "a/b", "a/b/c", "/a/b", "

Attenendomi alla documentazione, ho deciso di scegliere come stringhe non valide le due stringhe "a/b" e "a/b/c" in quanto non iniziano con il carattere slash ('/'). Per quanto riguarda le stringhe "/znode1", "/a/b" e "/a/b/c" esse dovrebbero essere stringhe valide in quanto soddisfano i requisiti descritti nella documentazione.

CLASSE: org.apache.zookeeper.server.DataTree.java

METODO: public void createNode (final String path, byte [] data, List<ACL> acl, long ephemeralOwner, int parentCVersion, long zxid, long time);

PARAMETRO	BOUNDARY-VALUES
path	null, "a", "node1/node2", "/a", "/abcd56", "
data	null, [], [b1, ..., bn]
acl	null, [], [acl1, ..., aclN]
ephemeralOwner	0, 23, 122, 0xFF00000000000001, 0x8000000000000000, -23

parentCVersion	-2, 0, 1, 3
zxid	-2, -1, 0, 1, 10000
Time	-2, -1, 0, 1, 300000

CLASSE: org.apache.zookeeper.server.DataTree.java

METODO: public void deleteNode (String path, long zxid);

PARAMETRO	BOUNDARY-VALUES
path	null, "", "/node1", "/a/b", "/a/b/c"
zxid	-1234567, -2, -1, 0, 1, 5, 10000, 10000000

IDENTIFICAZIONE CASI DI TEST.

CLASSE: org.apache.zookeeper.common.PathUtils.java

METODO: public static void validatePath (String path)

Inizialmente, nello studio di questo metodo ho adottato un approccio black-box. Verso la fine delle iterazioni, per rifinire la percentuale delle metriche, ho sfruttato il codice della classe. Per identificare il valore atteso, mi sono basato sul meccanismo delle eccezioni. Per capire se il metodo si comporta correttamente, verifico se entra in logica di errore per specifici input. Basandomi sulle osservazioni fatte in precedenza e sulla tecnica Boundary value, ho progettato i seguenti casi di test:

INPUT	VALORE ATTESO	DESCRIZIONE
{null}	Eccezione	Il metodo deve essere in grado di gestire correttamente il valore NULL.
“a/b/c”	Eccezione	Mi aspetto una eccezione in quanto non è un path valido. Il path fornito non inizia con lo slash (/)
“a/b”	Eccezione	-
“”	Eccezione	-
“/znode1”	Nessuna eccezione	Mi aspetto che il metodo non entri in logica di errore in quanto il valore di input rappresenta un path valido per uno ZNode.
“/a/b/c”	Nessuna eccezione	-
“/a/b”	Nessuna eccezione	-
“/a”	Nessuna eccezione	-

Implementando i casi di test descritti nella tabella precedente, non sono riuscito a coprire totalmente il metodo. Le metriche di adeguatezza per i test che ho scelto sono la STATEMENT COVERAGE e la CONDITION COVERAGE. Queste metriche mi vengono riportate direttamente da JaCoCo all'interno del report generato. Eseguendo i casi di test implementati, per quanto riguarda JaCoCo, ottengo i seguenti valori:



L'obiettivo che mi sono posto è quello di aumentare entrambe le percentuali mostrate nel report. Non essendo ancora soddisfatto della percentuale per le due metriche, ho deciso di effettuare una ulteriore iterazione prima di passare a Badua e a PIT andando ad aumentare la dimensione della test suite. Cercando ulteriori informazioni relative al progetto, ho ricavato l'informazione secondo cui non è possibile utilizzare il carattere “.” da solo per indicare un nodo lungo un percorso. Il motivo di ciò è che ZooKeeper non utilizza percorsi relativi. A tal proposito, ho ampliato la test suite aggiungendo i seguenti casi di test:

INPUT	VALORE ATTESO	DESCRIZIONE
“./a”	Eccezione	Mi aspetto una eccezione in quanto non è un path valido essendo un percorso relativo.
“/a/.b”	Eccezione	-
“a/b./”	Eccezione	-
“./.”	Eccezione	-

Implementando i casi di test descritti nella tabella precedente, sono riuscito ad aumentare entrambe le metriche come mostrato nella figura seguente:

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
validatePath(String)		65%		55%	18	28	11	35	0	1

Avendo raggiunto una percentuale superiore al 50%, ed essendo soddisfatto per questa seconda iterazione, ho deciso di iniziare ad applicare Badua e mutation testing. Per quanto riguarda Badua, i risultati sono mostrati nella [figura 2](#). Come è possibile osservare dal report, ci sono due coppie def-use per la variabile “path” che non sono state coperte. Ragionevolmente, questo potrebbe significare che ci sono dei casi particolari per il path di uno ZNode che non sono stati ancora considerati. Ho osservato che la maggior parte delle coppie def-use della variabile “reason” non vengono coperte dai casi di test progettati fino a questa iterazione. Ragionevolmente, la variabile “reason” può essere associata al motivo del fallimento del metodo e di conseguenza ci saranno casi particolari che ancora non sono stati testati per quanto riguarda la validità del path di uno ZNode. Inoltre, basandomi sulla copertura delle coppie che riguardano le variabili presenti all'interno del ciclo, posso osservare che il ciclo non è stato coperto in modo adeguato. Devo quindi ampliare la test suite con lo scopo di aumentare anche la copertura all'interno del ciclo.

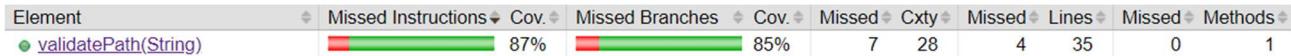
Utilizzando PIT, ottengo i risultati illustrati nella [figura 3](#). Osservo nella [figura 4](#) che sono state applicate ben 16 mutazioni e che poco meno della metà dei mutant sono stati identificati. Inoltre, osservo che in totale esattamente il 50% dei mutant sono stati uccisi. Un motivo per una percentuale così bassa potrebbe essere rappresentato dal fatto che molte righe di codice non sono state raggiunte dalla mia test suite. Mi sono posto l'obiettivo di ampliare la test suite con anche l'intento di insistere nella zona di codice mostrata nella figura 4 cercando di identificare un maggior numero di mutant.

Nell'iterazione successiva, ho provato ad aumentare le percentuali andando ad ampliare la test suite. Per ampliare la test suite, ho ricavato ulteriori informazioni dalle specifiche e da fonti collegate al progetto. Nella definizione del path di uno ZNode, il carattere “.” può essere utilizzato come parte di un altro nome. Inoltre, ho aggiunto dei casi di test che utilizzano la stringa “..” all'interno del path dello ZNode per coprire più casi riguardanti i percorsi relativi. Esistono dei caratteri Unicode che non possono essere inseriti all'interno del path dello ZNode. Di conseguenza, nei nuovi casi di test progettati ho considerato questi valori per vedere se il metodo riconosce il path come non valido. Infine, ho considerato il caso in cui il path immesso corrisponde alla radice dell'albero. La stringa “/” deve rappresentare un path valido in quanto corrisponde ad un elemento della struttura ad albero, per come è stato definito il modello dei dati nel progetto. Ho aggiunto i seguenti casi di test per ampliare la test suite:

INPUT	VALORE ATTESO	DESCRIZIONE
“/..”	Eccezione	Mi aspetto una eccezione in quanto non è un path valido essendo un percorso relativo.
“/a/..”	Eccezione	-
“/a/b/..”	Eccezione	-
“a/../b”	Eccezione	Mi aspetto una eccezione in quanto non è un path valido essendo un percorso relativo e non iniziando con il carattere slash (/)
“/u0001”	Eccezione	Mi aspetto una eccezione in quanto il path di uno ZNode non può contenere questo carattere perché non viene visualizzato correttamente oppure può essere visualizzato in modo confuso.
“/u0019”	Eccezione	-
“/u007F”	Eccezione	-
“/u009F”	Eccezione	-
“/ud800”	Eccezione	Mi aspetto una eccezione in quanto questo carattere non può essere utilizzato all'interno del path di uno ZNode.
“/uF8FF”	Eccezione	-
“/UFFF0”	Eccezione	-
“/uFFFF”	Eccezione	-
“/a/b/78/”	Eccezione	Mi aspetto una eccezione in quanto il path di uno ZNode non può terminare con il carattere slash ‘/’
“/abc.d”	Non eccezione	Mi aspetto che il metodo non lanci una eccezione in quanto il carattere “.” può essere utilizzato come parte di un altro nome.
“/.abcd”	Non eccezione	-
“/abcd.”	Non eccezione	-

"/"	Non eccezione	Mi aspetto che il metodo non lanci una eccezione perché questo path corrisponde alla root dell'albero.
-----	---------------	--

Con la nuova test suite sono riuscito ad aumentare notevolmente le percentuali delle metriche. Dal report di JaCoCo osservo i seguenti risultati:



Il report generato da Badua è mostrato nella [figura 5](#). Come si può vedere dalla figura, con la nuova test suite sono riuscito a coprire 27 coppie def-use in più rispetto a quelle coperte con la test suite precedente. Ho aumentato il numero di coppie coperte per quanto riguarda la variabile 'reason' e ho coperto tutte le coppie della variabile path che nella iterazione precedente non ero riuscito a coprire. Inoltre, osservo che con l'aggiunta di casi di test che vanno a coinvolgere maggiormente il ciclo, il numero di coppie def-use coperte per le variabili presenti all'interno ciclo è aumentato notevolmente.

Nella [figura 6](#), è riportato il report generato da PIT. Come si può osservare confrontando i report generati con PIT nelle ultime due iterazioni, aggiungendo i nuovi casi di test sono riuscito ad aumentare del 30% la mutation coverage. A differenza dell'iterazione precedente, per quanto riguarda la parte di codice soggetta alle 16 mutazioni, sono riuscito a identificare molti più mutanti dei 16 generati arrivando a soli due mutanti sopravvissuti. Il motivo è dovuto all'inserimento di casi di test che insistono soprattutto in quella zona di codice.

CLASSE: org.apache.zookeeper.server.DataTree.java

METODO: `public void createNode (final String path, byte [] data, List<ACL> acl, long ephemeralOwner, int parentCVersion, long zxid, long time);`

METODO: `public void deleteNode (String path, long zxid);`

Per quanto riguarda il parametro path, sono valide tutte le osservazioni fatte nello studio precedente per il metodo validatePath. Tuttavia, nel caso della creazione del nodo all'interno dell'albero, un path valido non è detto che sia corretto. La correttezza di un path dipende da come è stato costruito l'albero fino al momento precedente la creazione del nodo. Ad esempio, supponendo di voler creare il nodo "/a/b", esso ha un path valido ma sarà corretto solamente se in precedenza è stato inserito all'interno dell'albero il nodo parent "/a". Quindi un insieme di casi di test in cui il metodo dovrebbe fallire è rappresentato dalla creazione di un nodo che dichiara di avere un parent inesistente all'interno dell'albero. Dalla documentazione, non è consentito ai nodi effimeri di avere dei nodi figli. Di conseguenza, un altro insieme di casi di test per cui il metodo dovrebbe fallire è rappresentato dalla creazione di un nodo figlio di un nodo effimero. Nella prima iterazione, ho deciso di progettare per il metodo "createNode" i seguenti casi di test:

1. {"a", [b1, ..., bn], [], 0, 0, 0, 0}: creazione di un nodo non effimero con un path non valido. Mi aspetto che il nodo non venga inserito all'interno dell'albero.
2. {"node1/node2", [b1, ..., bn], null, 0xFF00000000000001L, 0, -1, 1}: creazione di un nodo TTL con un path non valido. Mi aspetto che il nodo non venga inserito all'interno dell'albero.
3. {"../node", null, OPEN_ACL_UNSAFE, 0x8000000000000000L, 0, 1, 300000}: creazione di un nodo container con un path non valido. Mi aspetto che il nodo non venga inserito all'interno dell'albero.
4. {"/a", [b1, ..., bn], CREATOR_ALL_ACL, 0, 3, 1, 1}: creazione di un nodo non effimero. Mi aspetto che il nodo venga inserito correttamente all'interno dell'albero. Il path del nodo è valido e corretto, è stata utilizzata una lista di ACL definita in ZooKeeper e il valore CVersion del nodo parent "/" deve essere settato a 3 perché prima dell'inserimento del nodo esso ha valore pari a zero.
5. {"/abc", [b1, ..., bn], OPEN_ACL_UNSAFE, 122, 0, 1, 1}: creazione di un nodo effimero standard. Mi aspetto che il nodo venga inserito all'interno dell'albero. Il valore CVersion del nodo parent "/" deve rimanere pari a zero.
6. {"/abcd56", [], CREATOR_ALL_ACL, 0, -2, 1, 1}: creazione di un nodo non effimero con array di dati vuoto e con un path valido e corretto. Mi aspetto che il nodo venga inserito all'interno dell'albero. Tuttavia, il valore di CVersion del nodo parent "/" non deve essere aggiornato perché il valore del parametro parentCVersion è strettamente minore di zero.
7. {"/a??9m", null, READ_ACL_UNSAFE, 122, 0, 1, 1}: creazione di un nodo effimero standard con array di dati 'null' e con un path valido e corretto. Mi aspetto che il nodo venga inserito all'interno dell'albero.
8. {"/a/b", [b1, ..., bn], OPEN_ACL_UNSAFE, 0, 0, 1, 1}: in questo caso di test, l'ambiente viene configurato in modo tale che il nodo "/a" sia un nodo effimero. Si tenta di creare un nodo non effimero "/a/b" che è figlio di un nodo effimero. La creazione del nodo non deve avvenire.

9. {"/a/b", [b1, ..., bn], READ_ACL_UNSAFE, 2, 0, 1, 1}: in questo caso di test, l'ambiente viene configurato in modo tale che il nodo "/a" sia un nodo effimero. Si tenta di creare un nodo effimero "/a/b" che è figlio di un nodo effimero. La creazione del nodo non deve avvenire.
10. {"/a/b", [b1, ..., bn], OPEN_ACL_UNSAFE, 0, 1, 10000, 1}: in questo test, l'ambiente viene configurato in modo tale che l'albero non contenga il nodo "/a". Di conseguenza, mi aspetto che il nodo non effimero "/a/b" non venga inserito all'interno dell'albero poiché il nodo genitore non esiste nella struttura dati.
11. {"/a/b", [b1, ..., bn], READ_ACL_UNSAFE, 23, 1, 1, 1}: in questo test, l'ambiente viene configurato in modo tale che l'albero non contenga il nodo "/a". Di conseguenza, mi aspetto che il nodo effimero standard "/a/b" non venga inserito all'interno dell'albero poiché il nodo genitore non esiste nella struttura dati.

Voglio sottolineare la differenza tra i casi di test 1-3 e i casi di test 10 e 11. Nei primi tre casi di test, l'errore nell'inserimento del nodo è dovuto ad un path che non è valido e di conseguenza non può essere corretto, indipendentemente dalla natura del nodo. Questo discorso è stato affrontato nello studio del metodo validatePath. Nel caso degli ultimi due casi di test, l'errore nell'inserimento del nodo è dovuto ad un path valido ma non corretto in quanto all'atto dell'inserimento ci si aspetta un determinato stato dell'albero che invece è diverso da quello attuale.

Eseguendo i test progettati, mi accorgo che i casi di test n°8/°9 falliscono poiché il nuovo nodo viene creato nonostante il padre sia un nodo effimero. Come riportato [nella documentazione](#), i nodi effimeri non possono avere figli. Inoltre, i casi di test n°1/°2/°3 fanno sì che il metodo fallisca per via di un errore diverso da quello che mi sarei aspettato. Andando a vedere il codice della classe DataTree.java, mi sono accorto che all'interno del metodo createNode non vengono fatti controlli sulla validità del path del nodo da aggiungere e sull'inserimento di un nodo avente come nodo genitore un nodo effimero. A questo punto, ho ipotizzato due diverse spiegazioni. La prima spiegazione è che all'interno del codice non sono stati implementati correttamente dei controlli e quindi all'interno della struttura ad albero possono esistere dei nodi il cui path non è concorde con la documentazione. Tuttavia, siccome questa classe ha un ruolo importante nel progetto ed è stata inserita numerose release precedenti, penso sia più probabile la seguente spiegazione: il metodo createNode della classe DataTree.java viene chiamato secondo un flusso di esecuzione ben preciso e non istanziando direttamente la classe ed invocando il metodo senza fare prima alcun controllo. A tal proposito, cercando un possibile flusso di esecuzione all'interno del progetto, ho notato che la classe ZooKeeper.java ha implementato l'operazione "create" in cui viene invocato il controllo sul path dello ZNode prima della sua effettiva creazione. Comunque sia, potrebbe essere un errore nella progettazione permettere di invocare il metodo "createNode" della classe DataTree.java senza prima aver eseguito tutti i controlli necessari. Non avendo a disposizione dei modelli del sistema (e.g., class diagram) che mi avrebbero permesso di osservare con più precisione le relazioni tra le varie classi del progetto, non posso trarre una conclusione definitiva e certa riguardante la problematica.

Per quanto riguarda il metodo "deleteNode", ho deciso di considerare i casi di test in cui viene richiesto di eliminare un nodo che non è presente all'interno dell'albero e i casi di test in cui viene richiesto di eliminare un nodo che è effettivamente presente nella struttura dati. Per testare l'eliminazione di un nodo che non è presente nell'albero, gestisco l'eccezione di tipo KeeperException.NoNodeException. Inoltre, osservo che per mantenere i dati consistenti è necessario controllare se il valore del parametro zxid passato in input è strettamente maggiore del valore di pzxid del nodo genitore. Il valore pzxid rappresenta lo zxid dell'ultima modifica fatta sui figli di uno ZNode. Nel caso in cui è strettamente maggiore, allora pzxid deve essere aggiornato. I casi di test progettati sono i seguenti:

1. {"/a", 10000}: per questo caso di test configuro l'albero in modo da non contenere il nodo "/a". Di conseguenza, mi aspetto che venga lanciata una eccezione di tipo KeeperException.NoNodeException in quanto il nodo da eliminare non è presente nella struttura dati.
2. {null, 0}: Mi aspetto che il metodo sia in grado di gestire il valore "null" lanciando una eccezione.
3. {"", -2}: poiché la stringa è vuota, mi aspetto che il metodo fallisca nell'eliminazione del nodo e che lanci una eccezione.

Questi primi tre casi di test si basano su un path di uno ZNode che non è presente all'interno dell'albero. A questo punto, ho progettato dei casi di test relativi all'eliminazione di ZNode effettivamente presenti all'interno dell'albero andando a gestire il corretto aggiornamento per il valore di pzxid. I casi di test progettati sono:

4. {"/a/b/node1/node2/c", -1}: Siccome il valore del parametro zxid è strettamente inferiore di zero, l'aggiornamento di pzxid (vale zero) non deve avvenire e il nodo non effimero "/a/b/node1/node2/c" viene eliminato correttamente dall'albero.

5. {"/a.b.c.d/e.f.g.h", 5}: Siccome il valore del parametro pzxid del nodo "/a.b.c.d" è pari a zero e il valore di zxid passato in input al metodo è cinque, allora l'eliminazione del nodo non effimero "/a.b.c.d/e.f.g.h" deve avvenire con successo e si deve verificare l'aggiornamento di pzxid.
6. {"/a/b", 0}: Il nodo non effimero "/a/b" viene eliminato dall'albero e l'aggiornamento non si verifica perché i valori di pzxid e zxid sono gli stessi.
7. {"/node1", 10000000}: Siccome il valore del parametro pzxid del nodo "/" è pari a zero e il valore di zxid passato in input al metodo è 10000000, allora l'eliminazione del nodo non effimero "/node1" deve avvenire con successo e si deve verificare l'aggiornamento di pzxid. Ho utilizzato un valore del parametro di zxid lontano dalla boundary.
8. {"/a/b/c", -1234567}: Siccome il valore del parametro zxid è strettamente inferiore di zero, l'aggiornamento di pzxid non deve avvenire e il nodo non effimero "/a/b/c" viene eliminato correttamente dall'albero. Ho utilizzato un valore del parametro di zxid lontano dalla boundary.

Dal report di JaCoCo ottengo i seguenti risultati.

Element	Missed Instructions	Cov.	Missed Branches	Cov.
● deleteNode(String, long)		63%		50%
● createNode(String, byte[], List, long, int, long, long, Stat)		80%		60%

Il report generato da Badua per i metodi createNode e deleteNode è mostrato nella [figura 7](#). Il report generato da PIT è mostrato nella [figura 8](#). Siccome la classe DataTree.java ha un elevato numero di metodi e il mio studio è focalizzato su solamente due di questi metodi, ho deciso di escludere dalla generazione dei mutanti tutti quanti i metodi diversi da deleteNode e createNode. Per fare ciò, sono andato sulla documentazione di PIT e ho cercato i TAG da inserire all'interno del profilo di PIT da me creato nel pom.xml. L'esecuzione del mutation testing è stata fatta con il valore di default del time-out. Tuttavia, aprendo il report di PIT osservo di non avere alcun mutante KILLED ma solamente tutti TIMED_OUT. Per ridurre il numero di TIMED_OUT, ho provato ad aumentare il valore del time-out di default utilizzando il tag "**timeoutConstant**". Tuttavia, cambiando in vari modi il valore di default del time-out non sono riuscito a diminuire il numero di TIMED_OUT e ottengo esattamente le stesse statistiche.

Nella seconda iterazione ho aggiunto ulteriori casi di test per aumentare le percentuali delle metriche. Per quanto riguarda il metodo createNode, ho aggiunto dei casi di test in cui provo a creare un nodo che già esiste all'interno dell'albero. Per gestire questo scenario, utilizzo l'eccezione KeeperException.NodeExistsException. Ovviamente, mi aspetto che il metodo lanci l'eccezione perché il nodo già è presente all'interno della struttura dati. Inoltre, aggiungo dei casi di test che coprono l'inserimento di un nodo container e di un nodo TTL. In entrambi i casi, l'inserimento ha successo in quanto il nodo non era presente nell'albero prima dell'inserimento. Per quanto riguarda il metodo deleteNode, ho aggiunto dei casi di test relativi alla rimozione di un nodo container, di un nodo TTL e di un nodo effimero, tutti presenti all'interno dell'albero. Inoltre, sempre per il metodo deleteNode, ho considerato il caso in cui viene eliminato un nodo che non ha il nodo genitore. Questo caso si può manifestare nel momento in cui viene eliminato prima il nodo genitore e poi si tenta di eliminare il nodo figlio. In questo ultimo caso di test, mi aspetto che il metodo lanci una eccezione perché non è in grado di trovare il nodo genitore.

Eseguendo la nuova test suite, sono riuscito ad aumentare il valore delle metriche. Dal report di JaCoCo ottengo i seguenti risultati:

Element	Missed Instructions	Cov.	Missed Branches	Cov.
● deleteNode(String, long)		77%		69%
● createNode(String, byte[], List, long, int, long, long, Stat)		87%		70%

Grazie ai nuovi casi di test implementati, sono riuscito a diminuire il numero delle coppie def-use per entrambi i metodi come si può vedere dal report generato da Badua nella [figura 9](#). Ad esempio, osservo che inserendo dei casi di test per il metodo createNode che prevedono la creazione di nodi TTL e container sono riuscito a coprire tutte le coppie per quanto riguarda la variabile "ephemeralType" e numerose altre coppie che riguardano variabili all'interno delle porzioni di codice legate all'inserimento di queste due tipologie di nodi. Inoltre, ho coperto anche l'ultima coppia per la variabile "children" grazie al caso di test che crea un nodo che già è presente all'interno dell'albero. Per il metodo deleteNode possono essere fatte considerazioni simili grazie ai casi di test che prevedono l'eliminazione di nodi TTL, container ed effimeri. Inoltre, sono riuscito a coprire la coppia def-use per quanto riguarda la variabile "parent" grazie al caso di test che prevede l'eliminazione di un nodo che non ha il nodo genitore. Per quanto riguarda l'esecuzione di PIT, osservo che l'ampliamento della test suite ha risolto i problemi che ho riscontrato nella iterazione precedente. I TIMED_OUT

che avevo nella iterazione precedente non sono più presenti all'interno del report. Il report di PIT è mostrato nella [figura 10](#). Anche in questa iterazione, ho dovuto aumentare il valore del time-out rispetto a quello di default.

PROFILO OPERAZIONALI.

BOOKKEEPER.

Nel progetto Apache BookKeeper, i casi di test che sono stati progettati ed implementati per i metodi delle due classi scelte non hanno riscontrato la presenza di alcun BUG. Nel metodo "read" della classe `BufferedChannel.java`, il fallimento del caso di test è dovuto ad una interpretazione scorretta del parametro "length" e non ad un BUG presente all'interno della classe. Di conseguenza, assumendo profili operazionali con probabilità di utilizzo uniforme rispetto all'insieme finale di test che è stato progettato per i metodi delle classi ottengo sempre una reliability pari ad uno. Questo ovviamente non implica il fatto che le classi sono prive di BUG.

ZOOKEEPER.

Nel progetto Apache ZooKeeper, i casi di test che sono stati progettati ed implementati relativi al metodo "validatePath" della classe `PathUtils.java` hanno tutti quanti avuto l'esito atteso. Di conseguenza, assumendo un profilo operazionale con probabilità di utilizzo uniforme rispetto alla test suite finale progettata posso dire che il valore della reliability è pari ad uno. Questo risultato non implica la mancanza di BUG.

Per quanto riguarda la classe `DataTree.java`, i casi di test progettati ed implementati relativi al metodo "deleteNode" hanno tutti quanti avuto l'esito atteso. Di conseguenza, in accordo alla spiegazione data per il metodo precedente, anche in questo caso ho una reliability pari ad uno. Il discorso relativo al metodo "createNode" è un po' più complesso. Riprendendo la problematica sollevata durante la prima iterazione di test, per tutto il processo di test è stata fatta l'assunzione secondo cui la creazione del nodo venisse fatta seguendo un preciso flusso di esecuzione che potrebbe anche essere quello descritto nella spiegazione relativa alla problematica. Tuttavia, nel caso in cui questa assunzione fosse errata, allora la mancanza di controllo per la creazione di un nodo figlio di un effimero e per la validità del path di uno ZNode rappresenterebbero un BUG. Assumendo un profilo operazionale con probabilità di utilizzo uniforme rispetto all'insieme dei casi di test progettati relativi al metodo "createNode" e considerando un totale di 14 casi di test, ottengo che la probabilità di ogni input risulta essere pari a 1 / 14. Di conseguenza, essendo cinque i casi di test che hanno portato il metodo ad avere una esecuzione differente da quella attesa, ottengo che la reliability è pari con una certa approssimazione a:

$$1 - PFD = 1 - 5/14 = 9/14 = 0.642857$$

Il profilo operazionale è mostrato nella [tabella](#) e ad ogni riga corrisponde una probabilità pari a circa 0.071429.

LINKS.

ZooKeeper Github: <https://github.com/luca-capotombolo/zookeeper>

BookKeeper Github: <https://github.com/luca-capotombolo/bookkeeper>

Fastjson Github: https://github.com/luca-capotombolo/fastjson_final_isw2

ZooKeeper SonarCloud: https://sonarcloud.io/project/overview?id=luca-capotombolo_zookeeper_isw2

BookKeeper SonarCloud: https://sonarcloud.io/project/overview?id=luca-capotombolo_bookkeeper

Fastjson SonarCloud: https://sonarcloud.io/summary/overall?id=luca-capotombolo_fastjson_final_isw2

FIGURE.**Figura 1a.**

```

package com.alibaba.json.bvt.parser;

import org.junit.Assert;

import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.parser.Feature;

import junit.framework.TestCase;

public class OrderedFieldTest extends TestCase {
    public void test_ordered_field() throws Exception {
        String text = "{\"id\":1001}";
        Model model = JSON.parseObject(text, Model.class, Feature.OrderedField);
        Assert.assertEquals(1001, model.getId());
        String text2 = JSON.toJSONString(model);
        Assert.assertEquals(text, text2);
    }

    public static interface Model {
        public int getId();
        public void setId(int value);
    }
}

```

Figura 2a.

```

public void test_cast_Integer() throws Exception {
    JSONObject json = new JSONObject();
    json.put("id", 1L);
    Assert.assertEquals(new Integer(1), json.getJSONObject("id", int.class));
}

public void test_cast_Integer_2() throws Exception {
    JSONObject json = new JSONObject();
    json.put("id", 1L);
    Assert.assertEquals(new Integer(1), json.getJSONObject("id", Integer.class));
}

```

Figura 3a.

```

luca-capotombolo
@Test
public void test_put_ID(){
    Assert.assertEquals(this.objectExpectedTestPutID, this.jsonObjectTestPutID.getJsonObject((String) this.objectInput1TestPutID, (Class<?>) this.objectInput2TestPutID));
}

```

Figura 4a.

```

luca-capotombolo *
@Test
public void test_2Gson(){
    Gson gson = new Gson();
    User userGson = gson.fromJson( json: "{" +this.objectInput3Test2+ ":" +this.objectInput5Test2+
    this.objectExpected1Test2 = userGson.id;
    this.objectExpected2Test2 = userGson.name;
    User user = TypeUtils.castToJavaBean(this.objectInput1Test2, this.objectInput2Test2);
    Assert.assertEquals(this.objectExpected1Test2, user.getId());
    Assert.assertEquals(this.objectExpected2Test2, user.getName());
}

```

Figura 5a.

● cast(Object, Class, ParserConfig) | 39% | 38%

Figura 6a.

```
public static <T> T cast(final Object obj, final Class<T> clazz, ParserConfig config) {
    if (obj == null) {
        if (clazz == int.class) {
            return (T) Integer.valueOf(0);
        } else if (clazz == long.class) {
            return (T) Long.valueOf(0);
        } else if (clazz == short.class) {
            return (T) Short.valueOf((short) 0);
        } else if (clazz == byte.class) {
            return (T) Byte.valueOf((byte) 0);
        } else if (clazz == float.class) {
            return (T) Float.valueOf(0);
        } else if (clazz == double.class) {
            return (T) Double.valueOf(0);
        } else if (clazz == boolean.class) {
            return (T) Boolean.FALSE;
        }
    }
    return null;
}
```

Figura 7a.

● cast(Object, Class, ParserConfig) | 49% | 51%

Figura 8a.

● castToBigDecimal(Object) | 16% | 14%

Figura 9a.

```
public static BigDecimal castToBigDecimal(Object value) {
    if (value == null) {
        return null;
    } else {
        if (value instanceof Float) {
            if (Float.isNaN((Float)value) || Float.isInfinite((Float)value)) {
                return null;
            }
        } else if (value instanceof Double) {
            if (Double.isNaN((Double)value) || Double.isInfinite((Double)value)) {
                return null;
            }
        } else {
            if (value instanceof BigDecimal) {
                return (BigDecimal)value;
            }

            if (value instanceof BigInteger) {
                return new BigDecimal((BigInteger)value);
            }

            if (value instanceof Map && ((Map)value).size() == 0) {
                return null;
            }
        }
    }

    String strVal = value.toString();
    if (strVal.length() != 0 && !strVal.equalsIgnoreCase("null")) {
```

Figura 10a.

● castToBigDecimal(Object) | 67% | 57%

Figura 11a.

● castToBigInteger(Object) | 14% | 14%

Figura 12a.

```
public static BigInteger castToBigInteger(Object value) {
    if (value == null) {
        return null;
    } else if (value instanceof Float) {
        Float floatValue = (Float)value;
        return !Float.isNaN(floatValue) && !Float.isInfinite(floatValue)
    } else if (value instanceof Double) {
        Double doubleValue = (Double)value;
        return !Double.isNaN(doubleValue) && !Double.isInfinite(doubleValue)
    } else if (value instanceof BigInteger) {
        return (BigInteger)value;
    }
```

Figura 13a.

● castToBigInteger(Object) | 55% | 46%

Figura a.**Mutations**

```

93 1. removed call to java/util/concurrent/atomic/AtomicLong::set → SURVIVED
97 1. changed conditional boundary → TIMED_OUT
2. negated conditional → TIMED_OUT
102 1. negated conditional → NO_COVERAGE
105 1. removed call to io/netty/util/ReferenceCountUtil::safeRelease → NO_COVERAGE
106 1. removed call to java/nio/channels/FileChannel::close → NO_COVERAGE
123 1. changed conditional boundary → TIMED_OUT
2. negated conditional → TIMED_OUT
124 1. Replaced integer subtraction with addition → SURVIVED
125 1. Replaced integer addition with subtraction → SURVIVED
126 1. Replaced integer addition with subtraction → KILLED
130 1. negated conditional → TIMED_OUT
131 1. removed call to org/apache/bookkeeper/bookie/BufferedChannel::flush → NO_COVERAGE
134 1. Replaced long addition with subtraction → SURVIVED
135 1. negated conditional → TIMED_OUT
137 1. changed conditional boundary → NO_COVERAGE
2. negated conditional → NO_COVERAGE
138 1. removed call to org/apache/bookkeeper/bookie/BufferedChannel::flush → NO_COVERAGE
143 1. negated conditional → SURVIVED
153 1. replaced long return with 0 for org/apache/bookkeeper/bookie/BufferedChannel::position → NO_COVERAGE
161 1. replaced long return with 0 for org/apache/bookkeeper/bookie/BufferedChannel::getFileChannelPosition → NO_COVERAGE
174 1. removed call to org/apache/bookkeeper/bookie/BufferedChannel::flush → NO_COVERAGE
188 1. negated conditional → NO_COVERAGE
189 1. removed call to org/apache/bookkeeper/bookie/BufferedChannel::flushAndForceWrite → NO_COVERAGE
203 1. negated conditional → TIMED_OUT
205 1. removed call to java/util/concurrent/atomic/AtomicLong::set → NO_COVERAGE
235 1. changed conditional boundary → NO_COVERAGE
2. negated conditional → NO_COVERAGE
237 1. removed call to java/util/concurrent/atomic/AtomicLong::set → NO_COVERAGE
241 1. removed call to java/nio/channels/FileChannel::force → NO_COVERAGE
242 1. replaced long return with 0 for org/apache/bookkeeper/bookie/BufferedChannel::forceWrite → NO_COVERAGE
248 1. changed conditional boundary → KILLED
2. negated conditional → KILLED
250 1. changed conditional boundary → KILLED
2. negated conditional → KILLED
3. negated conditional → KILLED
251 1. Replaced long subtraction with addition → SURVIVED
252 1. Replaced integer subtraction with addition → KILLED
254 1. negated conditional → TIMED_OUT
259 1. Replaced long addition with subtraction → KILLED
260 1. Replaced integer subtraction with addition → KILLED
1. changed conditional boundary → NO_COVERAGE
261 2. negated conditional → NO_COVERAGE
3. negated conditional → NO_COVERAGE
1. changed conditional boundary → TIMED_OUT
2. Replaced long addition with subtraction → TIMED_OUT
265 3. negated conditional → TIMED_OUT
4. negated conditional → TIMED_OUT
5. changed conditional boundary → NO_COVERAGE
266 1. Replaced long subtraction with addition → NO_COVERAGE
267 1. Replaced integer subtraction with addition → NO_COVERAGE
269 1. Replaced long addition with subtraction → NO_COVERAGE
270 1. Replaced integer subtraction with addition → NO_COVERAGE
1. changed conditional boundary → NO_COVERAGE
277 2. negated conditional → NO_COVERAGE
283 1. Replaced long subtraction with addition → SURVIVED
2. replaced int return with 0 for org/apache/bookkeeper/bookie/BufferedChannel::read → KILLED
288 1. removed call to org/apache/bookkeeper/bookie/BufferedReadChannel::clear → NO_COVERAGE
293 1. replaced int return with 0 for org/apache/bookkeeper/bookie/BufferedChannel::getNumOfBytesInWriteBuffer → NO_COVERAGE
297 1. replaced long return with 0 for org/apache/bookkeeper/bookie/BufferedChannel::getUnpersistedBytes → NO_COVERAGE

```

Pit Test Coverage Report

Package Summary

org.apache.bookkeeper.bookie

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
1	43% <div style="width: 43%; background-color: #90EE90; height: 10px;"></div>	37% <div style="width: 37%; background-color: #FFB6C1; height: 10px;"></div>	76% <div style="width: 76%; background-color: #90EE90; height: 10px;"></div>

Breakdown by Class

Name	Line Coverage	Mutation Coverage	Test Strength
BufferedChannel.java	43% <div style="width: 43%; background-color: #90EE90; height: 10px;"></div>	37% <div style="width: 37%; background-color: #FFB6C1; height: 10px;"></div>	76% <div style="width: 76%; background-color: #90EE90; height: 10px;"></div>

Figura b.

```

▼<method name="write" desc="(Lio/netty/buffer/ByteBuf;)V">
<du var="this" def="119" use="134" covered="1"/>
<du var="this" def="119" use="135" target="136" covered="0"/>
<du var="this" def="119" use="135" target="142" covered="1"/>
<du var="this" def="119" use="144" covered="0"/>
<du var="this" def="119" use="136" covered="0"/>
<du var="this" def="119" use="137" target="138" covered="0"/>
<du var="this" def="119" use="137" target="142" covered="0"/>
<du var="this" def="119" use="138" covered="0"/>
<du var="this" def="119" use="124" covered="1"/>
<du var="this" def="119" use="125" covered="1"/>
<du var="this" def="119" use="130" target="131" covered="0"/>
<du var="this" def="119" use="130" target="133" covered="1"/>
<du var="this" def="119" use="131" covered="0"/>
<du var="src" def="119" use="124" covered="1"/>
<du var="src" def="119" use="125" covered="1"/>
<du var="this.writeBuffer" def="119" use="124" covered="1"/>
<du var="this.writeBuffer" def="119" use="125" covered="1"/>
<du var="this.writeBuffer" def="119" use="130" targets="131" covered="0"/>
<du var="this.writeBuffer" def="119" use="130" targets="133" covered="1"/>
<du var="this.position" def="119" use="134" covered="1"/>
<du var="this.doRegularFlushes" def="119" use="135" target="136" covered="0"/>
<du var="this.doRegularFlushes" def="119" use="135" target="142" covered="0"/>
<du var="this.unpersistedBytes" def="119" use="136" covered="0"/>
<du var="this.unpersistedBytes" def="119" use="137" target="138" covered="0"/>
<du var="this.unpersistedBytesBound" def="119" use="137" target="138" covered="0"/>
<du var="this.unpersistedBytesBound" def="119" use="137" target="142" covered="0"/>
<du var="copied" def="119" use="123" target="124" covered="1"/>
<du var="copied" def="119" use="134" covered="1"/>
<du var="copied" def="119" use="136" covered="0"/>
<du var="copied" def="119" use="124" covered="1"/>
<du var="copied" def="119" use="125" covered="1"/>
<du var="copied" def="119" use="126" covered="1"/>
<du var="shouldForceWrite" def="120" use="143" target="144" covered="0"/>
<du var="shouldForceWrite" def="120" use="143" target="146" covered="1"/>
<du var="len" def="122" use="123" target="124" covered="1"/>
<du var="len" def="122" use="123" target="134" covered="1"/>
<du var="copied" def="126" use="123" target="124" covered="0"/>
<du var="copied" def="126" use="123" target="134" covered="1"/>
<du var="copied" def="126" use="134" covered="1"/>
<du var="copied" def="126" use="136" covered="0"/>
<du var="copied" def="126" use="124" covered="0"/>
<du var="copied" def="126" use="125" covered="1"/>
<du var="copied" def="126" use="126" covered="0"/>
<du var="shouldForceWrite" def="139" use="143" target="144" covered="0"/>
<du var="shouldForceWrite" def="139" use="143" target="146" covered="0"/>
<counter type="DU" missed="24" covered="24"/>
<counter type="METHOD" missed="0" covered="1"/>
</method>
▼<method name="read" desc="(Lio/netty/buffer/ByteBuf;JI)I">
<du var="this" def="247" use="250" target="250" covered="1"/>
<du var="this" def="247" use="250" target="261" covered="0"/>
<du var="this" def="247" use="261" target="261" covered="0"/>
<du var="this" def="247" use="261" target="265" covered="0"/>
<du var="this" def="247" use="265" target="265" covered="0"/>
<du var="this" def="247" use="273" target="273" covered="0"/>
<du var="this" def="247" use="273" covered="0"/>
<du var="this" def="247" use="275" covered="0"/>
<du var="this" def="247" use="280" covered="0"/>
<du var="this" def="247" use="265" target="266" covered="0"/>
<du var="this" def="247" use="265" target="273" covered="0"/>
<du var="this" def="247" use="266" covered="0"/>
<du var="this" def="247" use="267" covered="0"/>
<du var="this" def="247" use="268" covered="0"/>
<du var="this" def="247" use="261" target="263" covered="0"/>
<du var="this" def="247" use="261" target="265" covered="0"/>
<du var="this" def="247" use="250" target="251" covered="1"/>
<du var="this" def="247" use="250" target="261" covered="0"/>
<du var="this" def="247" use="251" covered="1"/>
<du var="this" def="247" use="252" covered="1"/>
<du var="this" def="247" use="258" covered="1"/>
<du var="dest" def="247" use="267" covered="0"/>
<du var="dest" def="247" use="268" covered="0"/>
<du var="dest" def="247" use="252" covered="1"/>
<du var="pos" def="247" use="258" covered="1"/>
<du var="pos" def="247" use="283" covered="1"/>
<du var="pos" def="247" use="265" target="265" covered="0"/>
<du var="pos" def="247" use="265" target="273" covered="0"/>
<du var="pos" def="247" use="273" covered="0"/>
<du var="pos" def="247" use="247" target="266" covered="0"/>
<du var="pos" def="247" use="265" target="273" covered="0"/>
<du var="pos" def="247" use="266" covered="0"/>
<du var="pos" def="247" use="269" covered="0"/>
<du var="pos" def="247" use="261" target="263" covered="0"/>
<du var="pos" def="247" use="261" target="265" covered="0"/>
<du var="pos" def="247" use="250" target="251" covered="1"/>
<du var="pos" def="247" use="250" target="261" covered="0"/>
<du var="pos" def="247" use="251" covered="1"/>
<du var="pos" def="247" use="259" covered="1"/>
<du var="length" def="247" use="248" target="250" covered="1"/>
<du var="length" def="247" use="248" target="283" covered="1"/>
<du var="length" def="247" use="270" covered="0"/>
<du var="length" def="247" use="260" covered="1"/>
<du var="this.writeBuffer" def="247" use="250" target="250" covered="1"/>
<du var="this.writeBuffer" def="247" use="250" target="261" covered="0"/>
<du var="this.writeBuffer" def="247" use="261" target="261" covered="0"/>
<du var="this.writeBuffer" def="247" use="261" target="265" covered="0"/>
<du var="this.writeBuffer" def="247" use="252" covered="1"/>
<du var="this.writeBuffer" def="247" use="258" covered="1"/>
<du var="this.writeBufferStartPosition" def="247" use="261" target="263" covered="0"/>
<du var="this.writeBufferStartPosition" def="247" use="261" target="265" covered="0"/>
<du var="this.writeBufferStartPosition" def="247" use="250" target="251" covered="1"/>
<du var="this.writeBufferStartPosition" def="247" use="250" target="261" covered="0"/>
<du var="this.readBufferStartPosition" def="247" use="251" covered="1"/>
<du var="this.readBufferStartPosition" def="247" use="265" target="265" covered="0"/>
<du var="this.readBufferStartPosition" def="247" use="265" target="273" covered="0"/>
<du var="this.readBufferStartPosition" def="247" use="266" covered="0"/>
<du var="this.readBuffer" def="247" use="275" covered="0"/>

```

```

<du var="this.readBuffer" def="247" use="280" covered="0"/>
<du var="this.readBuffer" def="247" use="265" target="266" covered="0"/>
<du var="this.readBuffer" def="247" use="267" covered="0"/>
<du var="this.readBuffer" def="247" use="268" covered="0"/>
<du var="this.fileChannel" def="247" use="275" covered="0"/>
<du var="prevPos" def="247" use="283" covered="1"/>
<du var="positionInBuffer" def="251" use="258" covered="1"/>
<du var="bytesToCopy" def="252" use="254" target="255" covered="1"/>
<du var="bytesToCopy" def="252" use="254" target="258" covered="1"/>
<du var="bytesToCopy" def="252" use="258" covered="1"/>
<du var="bytesToCopy" def="252" use="259" covered="1"/>
<du var="bytesToCopy" def="252" use="260" covered="1"/>
<du var="pos" def="259" use="283" covered="1"/>
<du var="pos" def="259" use="265" target="265" covered="0"/>
<du var="pos" def="259" use="265" target="273" covered="0"/>
<du var="pos" def="259" use="273" covered="0"/>
<du var="pos" def="259" use="265" target="266" covered="0"/>
<du var="pos" def="259" use="265" target="273" covered="0"/>
<du var="pos" def="259" use="266" covered="0"/>
<du var="pos" def="259" use="269" covered="0"/>
<du var="pos" def="259" use="261" target="263" covered="0"/>
<du var="pos" def="259" use="261" target="265" covered="0"/>
<du var="pos" def="259" use="250" target="251" covered="1"/>
<du var="pos" def="259" use="250" target="261" covered="0"/>
<du var="pos" def="259" use="251" covered="1"/>
<du var="pos" def="259" use="259" covered="0"/>
<du var="length" def="260" use="248" target="250" covered="1"/>
<du var="length" def="260" use="248" target="283" covered="1"/>
<du var="length" def="260" use="270" covered="0"/>
<du var="length" def="260" use="260" covered="0"/>
<du var="pos" def="269" use="283" covered="0"/>
<du var="pos" def="269" use="265" target="265" covered="0"/>
<du var="pos" def="269" use="265" target="273" covered="0"/>
<du var="pos" def="269" use="266" target="266" covered="0"/>
<du var="pos" def="269" use="265" target="273" covered="0"/>
<du var="pos" def="269" use="266" covered="0"/>
<du var="pos" def="269" use="269" covered="0"/>
<du var="pos" def="269" use="261" target="263" covered="0"/>
<du var="pos" def="269" use="261" target="265" covered="0"/>
<du var="pos" def="269" use="250" target="251" covered="0"/>
<du var="pos" def="269" use="250" target="261" covered="0"/>
<du var="pos" def="269" use="251" covered="0"/>
<du var="pos" def="269" use="259" covered="0"/>
<du var="length" def="270" use="248" target="250" covered="0"/>
<du var="length" def="270" use="248" target="283" covered="0"/>
<du var="length" def="270" use="270" covered="0"/>
<du var="length" def="270" use="260" covered="0"/>
<du var="this.readBufferStartPosition" def="273" use="265" target="265" covered="0"/>
<du var="this.readBufferStartPosition" def="273" use="265" target="273" covered="0"/>
<du var="this.readBufferStartPosition" def="273" use="265" target="266" covered="0"/>
<du var="this.readBufferStartPosition" def="273" use="265" target="273" covered="0"/>
<du var="this.readBufferStartPosition" def="273" use="266" covered="0"/>
<du var="readBytes" def="275" use="277" target="278" covered="0"/>
<du var="readBytes" def="275" use="277" target="280" covered="0"/>
<du var="readBytes" def="275" use="280" covered="0"/>
<counter type="DU" missed="87" covered="31"/>
<counter type="METHOD" missed="0" covered="1"/>
</method>

```

Figura c.

```

261 } else if (writeBuffer == null && writeBufferStartPosition.get() <= pos) {
262     // here we reach the end
263     break;
264     // first check if there is anything we can grab from the readBuffer
265 } else if (readBufferStartPosition <= pos && pos < readBufferStartPosition + readBuffer.writerIndex()) {
266     int positionInBuffer = (int) (pos - readBufferStartPosition);
267     int bytesToCopy = Math.min(readBuffer.writerIndex() - positionInBuffer, dest.writableBytes());
268     dest.writeBytes(readBuffer, positionInBuffer, bytesToCopy);
269     pos += bytesToCopy;
270     length -= bytesToCopy;
271     // let's read it
272 } else {
273     readBufferStartPosition = pos;
274
275     int readBytes = fileChannel.read(readBuffer.internalNioBuffer(0, readCapacity),
276                                     readBufferStartPosition);
277     if (readBytes <= 0) {
278         throw new IOException("Reading from filechannel returned a non-positive value. Short read.");
279     }
280     readBuffer.writerIndex(readBytes);
281 }
282 }
```

Figura d.

```

<du var="this.readBuffer" def="247" use="280" covered="1"/>
<du var="this.readBuffer" def="247" use="265" target="266" covered="1"/>
<du var="this.readBuffer" def="247" use="265" target="273" covered="1"/>
<du var="this.readBuffer" def="247" use="267" covered="1"/>
<du var="this.readBuffer" def="247" use="268" covered="1"/>
<du var="this.fileChannel" def="247" use="275" covered="1"/>
<du var="prePos" def="247" use="283" covered="1"/>
<du var="positionInBuffer" def="251" use="258" covered="1"/>
<du var="bytesToCopy" def="252" use="254" target="255" covered="1"/>
<du var="bytesToCopy" def="252" use="254" target="258" covered="1"/>
<du var="bytesToCopy" def="252" use="258" covered="1"/>
<du var="bytesToCopy" def="252" use="259" covered="1"/>
<du var="bytesToCopy" def="259" use="260" covered="1"/>
<du var="bytesToCopy" def="259" use="283" covered="1"/>
<du var="pos" def="259" use="265" target="265" covered="0"/>
<du var="pos" def="259" use="265" target="273" covered="0"/>
<du var="pos" def="259" use="273" covered="0"/>
<du var="pos" def="259" use="265" target="266" covered="0"/>
<du var="pos" def="259" use="265" target="273" covered="0"/>
<du var="pos" def="259" use="266" covered="0"/>
<du var="pos" def="259" use="269" covered="0"/>
<du var="pos" def="259" use="261" target="263" covered="0"/>
<du var="pos" def="259" use="261" target="265" covered="0"/>
<du var="pos" def="259" use="250" target="251" covered="1"/>
<du var="pos" def="259" use="251" covered="1"/>
<du var="pos" def="259" use="259" covered="0"/>
<du var="length" def="260" use="248" target="250" covered="1"/>
<du var="length" def="260" use="248" target="283" covered="1"/>
<du var="length" def="260" use="270" covered="0"/>
<du var="length" def="260" use="260" covered="0"/>
<du var="pos" def="269" use="283" covered="0"/>
<du var="pos" def="269" use="265" target="265" covered="0"/>
<du var="pos" def="269" use="265" target="273" covered="0"/>
<du var="pos" def="269" use="273" covered="0"/>
<du var="pos" def="269" use="265" target="266" covered="0"/>
<du var="pos" def="269" use="265" target="273" covered="0"/>
<du var="pos" def="269" use="266" covered="0"/>
<du var="pos" def="269" use="269" covered="0"/>
<du var="pos" def="269" use="261" target="263" covered="0"/>
<du var="pos" def="269" use="261" target="265" covered="0"/>
<du var="pos" def="269" use="250" target="251" covered="1"/>
<du var="pos" def="269" use="250" target="261" covered="0"/>
<du var="pos" def="269" use="250" target="262" covered="0"/>
<du var="pos" def="269" use="251" covered="1"/>
<du var="pos" def="269" use="259" covered="1"/>
<du var="length" def="270" use="248" target="250" covered="1"/>
<du var="length" def="270" use="248" target="283" covered="0"/>
<du var="length" def="270" use="270" covered="0"/>
<du var="length" def="270" use="260" covered="0"/>
<du var="readBufferStartPosition" def="273" use="265" target="265" covered="1"/>
<du var="this.readBufferStartPosition" def="273" use="265" target="273" covered="0"/>
<du var="this.readBufferStartPosition" def="273" use="265" target="266" covered="1"/>
<du var="this.readBufferStartPosition" def="273" use="265" target="273" covered="0"/>
<du var="this.readBufferStartPosition" def="273" use="266" covered="1"/>
<du var="readBytes" def="275" use="277" target="278" covered="0"/>
<du var="readBytes" def="275" use="277" target="280" covered="1"/>
<du var="readBytes" def="275" use="280" covered="1"/>
<counter type="DU" missed="44" covered="74"/>
<counter type="METHOD" missed="0" covered="1"/>

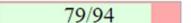
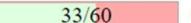
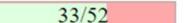
```

Figura e.

Pit Test Coverage Report

Package Summary

org.apache.bookkeeper.bookie

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
1	84%  79/94	55%  33/60	63%  33/52

Breakdown by Class

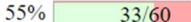
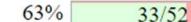
Name	Line Coverage	Mutation Coverage	Test Strength
BufferedChannel.java	84%  79/94	55%  33/60	63%  33/52

Figura f.

```

    @Test
    public void testRead() throws IOException {

        int expected;
        int a = this.numberByteWritten-this.pos;
        if((a<=this.sizeReadBuf && a >= this.length) || (a > this.sizeReadBuf && this.sizeReadBuf >= this.length)) {
            int n = this.bufferedChannel.read(readBuf, this.pos, this.length);
            if (a <= this.sizeReadBuf && a >= this.length)
                expected = a;
            else
                expected = this.sizeReadBuf;
            Assert.assertEquals(expected, n);
        }else {
            Exception error = null;
            try{
                this.bufferedChannel.read(readBuf, this.pos, this.length);
            }catch (Exception e){
                error = e;
            }
            Assert.assertNotNull(error);
        }
    }
}

```

Figura g.

Pit Test Coverage Report

Package Summary

`org.apache.bookkeeper.bookie`

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
1	3%	23%	100%

Breakdown by Class

Name	Line Coverage	Mutation Coverage	Test Strength
<code>BookieImpl.java</code>	3%	23%	100%

Mutations

226	1. changed conditional boundary → NO_COVERAGE
227	2. negated conditional → NO_COVERAGE
231	1. negated conditional → NO_COVERAGE
231	1. removed call to org/apache/commons/lang3/mutable/MutableBoolean::setValue → NO_COVERAGE
255	1. changed conditional boundary → KILLED
255	2. negated conditional → TIMED_OUT
255	3. negated conditional → KILLED
257	1. replaced return value with null for org/apache/bookkeeper/bookie/BookieImpl::getBookieAddress → NO_COVERAGE
261	1. negated conditional → TIMED_OUT
267	1. negated conditional → TIMED_OUT
273	1. negated conditional → TIMED_OUT
275	1. negated conditional → NO_COVERAGE
288	1. negated conditional → TIMED_OUT
289	1. negated conditional → NO_COVERAGE
297	1. replaced return value with null for org/apache/bookkeeper/bookie/BookieImpl::getBookieAddress → TIMED_OUT
305	1. replaced return value with null for org/apache/bookkeeper/bookie/BookieImpl::getIndexDirsManager → NO_COVERAGE
322	1. changed conditional boundary → NO_COVERAGE
322	2. negated conditional → NO_COVERAGE
325	1. replaced return value with null for org/apache/bookkeeper/bookie/BookieImpl::getCurrentDirectories → NO_COVERAGE
404	1. removed call to org/apache/bookkeeper/bookie/BookieImpl::checkEnvironment → NO_COVERAGE
407	1. removed call to org/apache/bookkeeper/bookie/StateManager::setShutdownHandler → NO_COVERAGE
407	2. removed call to org/apache/bookkeeper/bookie/BookieImpl::triggerBookieShutdown → NO_COVERAGE
413	1. negated conditional → NO_COVERAGE
418	1. removed call to org/apache/bookkeeper/bookie/LedgerDirsMonitor::init → NO_COVERAGE
421	1. negated conditional → NO_COVERAGE
430	1. changed conditional boundary → NO_COVERAGE
430	2. negated conditional → NO_COVERAGE
454	1. negated conditional → NO_COVERAGE
454	2. negated conditional → NO_COVERAGE
477	1. removed call to org/apache/bookkeeper/bookie/LedgerStorage::setStateManager → NO_COVERAGE
478	1. removed call to org/apache/bookkeeper/bookie/LedgerStorage::setCheckpointSource → NO_COVERAGE
479	1. removed call to org/apache/bookkeeper/bookie/LedgerStorage::setCheckpointer → NO_COVERAGE
479	1. changed conditional boundary → NO_COVERAGE
603	2. negated conditional → NO_COVERAGE
603	3. replaced boolean return with true for org/apache/bookkeeper/bookie/BookieImpl::lambda\$replay\$2 → NO_COVERAGE

Figura h.**Mutations**

226	1. changed conditional boundary → NO_COVERAGE
227	2. negated conditional → NO_COVERAGE
231	1. removed call to org/apache/commons/lang3/mutable/MutableBoolean::setValue → NO_COVERAGE
255	1. changed conditional boundary → KILLED
255	2. negated conditional → KILLED
255	3. negated conditional → KILLED
257	1. replaced return value with null for org/apache/bookkeeper/bookie/BookieImpl::getBookieAddress → KILLED
261	1. negated conditional → KILLED
267	1. negated conditional → KILLED
273	1. negated conditional → KILLED
275	1. negated conditional → SURVIVED
288	1. negated conditional → KILLED
289	1. negated conditional → NO_COVERAGE
297	1. replaced return value with null for org/apache/bookkeeper/bookie/BookieImpl::getBookieAddress → KILLED
305	1. replaced return value with null for org/apache/bookkeeper/bookie/BookieImpl::getIndexDirsManager → NO_COVERAGE

Pit Test Coverage Report

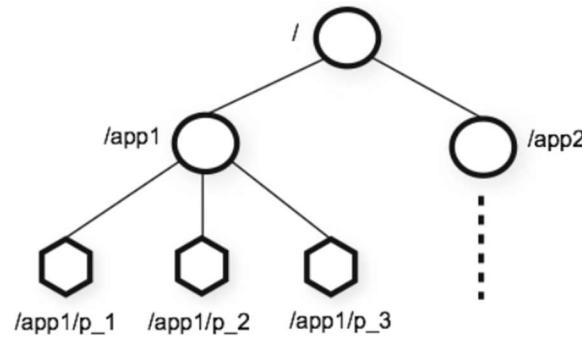
Package Summary

org.apache.bookkeeper.bookie

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
1	4% 21/469	26% 9/35	90% 9/10

Breakdown by Class

Name	Line Coverage	Mutation Coverage	Test Strength
BookieImpl.java	4% 21/469	26% 9/35	90% 9/10

Figura 1.**Figura 2.**

```

▼<class name="org/apache/zookeeper/common/PathUtils">
  ▶<method name="validatePath" desc="(Ljava/lang/String;Z)V">
    ...
  </method>
  ▼<method name="validatePath" desc="(Ljava/lang/String;)V">
    <du var="path" def="42" use="42" target="43" covered="1"/>
    <du var="path" def="42" use="42" target="45" covered="1"/>
    <du var="path" def="42" use="45" target="46" covered="1"/>
    <du var="path" def="42" use="45" target="48" covered="1"/>
    <du var="path" def="42" use="48" target="49" covered="1"/>
    <du var="path" def="42" use="48" target="51" covered="1"/>
    <du var="path" def="42" use="51" target="52" covered="0"/>
    <du var="path" def="42" use="51" target="54" covered="1"/>
    <du var="path" def="42" use="54" target="55" covered="0"/>
    <du var="path" def="42" use="54" target="58" covered="1"/>
    <du var="path" def="42" use="60" covered="1"/>
    <du var="path" def="42" use="91" covered="1"/>
    <du var="reason" def="58" use="90" target="91" covered="0"/>
    <du var="reason" def="58" use="90" target="93" covered="1"/>
    <du var="reason" def="58" use="91" covered="0"/>
    <du var="lastc" def="59" use="71" target="72" covered="0"/>
    <du var="lastc" def="59" use="71" target="76" covered="1"/>
    <du var="lastc" def="59" use="68" target="69" covered="0"/>
    <du var="lastc" def="59" use="68" target="71" covered="0"/>
    <du var="chars" def="60" use="62" target="63" covered="1"/>
    <du var="chars" def="60" use="62" target="90" covered="1"/>
    <du var="chars" def="60" use="63" covered="1"/>
    <du var="chars" def="60" use="62" covered="1"/>
    <du var="chars" def="60" use="77" target="77" covered="1"/>
    <du var="chars" def="60" use="77" target="62" covered="0"/>
    <du var="chars" def="60" use="77" target="77" covered="1"/>
    <du var="chars" def="60" use="77" target="78" covered="1"/>
    <du var="chars" def="60" use="77" target="78" covered="1"/>
    <du var="chars" def="60" use="77" target="62" covered="0"/>
    <du var="chars" def="60" use="72" target="72" covered="0"/>
    <du var="chars" def="60" use="72" target="62" covered="0"/>
    <du var="chars" def="60" use="72" target="72" covered="0"/>
    <du var="chars" def="60" use="72" target="73" covered="0"/>
    <du var="chars" def="60" use="72" target="73" covered="0"/>
    <du var="chars" def="60" use="72" target="62" covered="0"/>
    <du var="i" def="62" use="62" target="63" covered="1"/>
    <du var="i" def="62" use="62" target="90" covered="0"/>
    <du var="i" def="62" use="63" covered="1"/>
    <du var="i" def="62" use="62" covered="1"/>
    <du var="i" def="62" use="62" covered="1"/>
    <du var="i" def="62" use="85" covered="0"/>
    <du var="i" def="62" use="77" target="77" covered="1"/>
  
```


Figura 3.

Pit Test Coverage Report

Package Summary

org.apache.zookeeper.common

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
1	57%	50%	77%

Breakdown by Class

Name	Line Coverage	Mutation Coverage	Test Strength
PathUtils.java	57%	50%	77%

```

1 public static void validatePath(String path) throws IllegalArgumentException {
2     if (path == null) {
3         throw new IllegalArgumentException("Path cannot be null");
4     }
5     if (path.length() == 0) {
6         throw new IllegalArgumentException("Path length must be > 0");
7     }
8     if (path.charAt(0) != '/') {
9         throw new IllegalArgumentException("Path must start with / character");
10    }
11    if (path.length() == 1) { // done checking - it's the root
12        return;
13    }
14    if (path.charAt(path.length() - 1) == '/') {
15        throw new IllegalArgumentException("Path must not end with / character");
16    }
17
18    String reason = null;
19    char lastc = '/';
20    char[] chars = path.toCharArray();
21    char c;
22    for (int i = 1; i < chars.length; lastc = chars[i], i++) {
23        c = chars[i];
24
25        if (c == 0) {
26            reason = "null character not allowed @" + i;
27            break;
28        } else if (c == '/' && lastc == '/') {
29            reason = "empty node name specified @" + i;
30            break;
31        } else if (c == '.' && lastc == '.') {
32            if (chars[i - 2] == '/' && ((i + 1 == chars.length) || chars[i + 1] == '/')) {
33                reason = "relative paths not allowed @" + i;
34                break;
35            }
36        } else if (c == '.') {
37            if (chars[i - 1] == '/' && ((i + 1 == chars.length) || chars[i + 1] == '/')) {
38                reason = "relative paths not allowed @" + i;
39                break;
40            }
41        } else if (c > '\u0000' && c <= '\u001f'
42                  || c >= '\u007f' && c <= '\u009f'
43                  || c >= '\ud800' && c <= '\uf8ff'
44                  || c >= '\ufff0' && c <= '\uffff') {
45            reason = "invalid character @" + i;
46            break;
47        }
48    }
49
50    if (reason != null) {
51        throw new IllegalArgumentException("Invalid path string \\" + path + "\\" caused by " + reason);
52    }
53}

```

Figura 4.

```

16 } else if (c > '\u0000' && c <= '\u001f'
    || c >= '\u007f' && c <= '\u009F'
    || c >= '\ud800' && c <= '\uf8ff'
    || c >= '\ufff0' && c <= '\uffff') {
    reason = "invalid character @" + i;
    break;
}

```

Figura 5.

```

▼<class name="org/apache/zookeeper/common/PathUtils">
  ▼<method name="validatePath" desc="(Ljava/lang/String;)V">
    <du var="path" def="33" use="33" covered="0"/>
    <du var="isSequential" def="33" use="33" target="33" covered="0"/>
    <du var="isSequential" def="33" use="33" target="33" covered="0"/>
    <counter type="DU" missed="3" covered="0"/>
    <counter type="METHOD" missed="1" covered="0"/>
  </method>
  ▼<method name="validatePath" desc="(Ljava/lang/String;)V">
    <du var="path" def="42" use="42" target="43" covered="1"/>
    <du var="path" def="42" use="42" target="45" covered="1"/>
    <du var="path" def="42" use="45" target="46" covered="1"/>
    <du var="path" def="42" use="45" target="48" covered="1"/>
    <du var="path" def="42" use="48" target="49" covered="1"/>
    <du var="path" def="42" use="48" target="51" covered="1"/>
    <du var="path" def="42" use="51" target="52" covered="1"/>
    <du var="path" def="42" use="51" target="54" covered="1"/>
    <du var="path" def="42" use="54" targets="55" covered="1"/>
    <du var="path" def="42" use="54" target="58" covered="1"/>
    <du var="path" def="42" use="60" covered="1"/>
    <du var="path" def="42" use="91" covered="1"/>
    <du var="reason" def="58" use="90" target="91" covered="0"/>
    <du var="reason" def="58" use="90" target="93" covered="1"/>
    <du var="reason" def="58" use="91" covered="0"/>
    <du var="lastc" def="59" use="71" target="72" covered="0"/>
    <du var="lastc" def="59" use="71" target="76" covered="1"/>
    <du var="lastc" def="59" use="68" target="69" covered="0"/>
    <du var="lastc" def="59" use="68" target="71" covered="0"/>
    <du var="chars" def="60" use="62" target="63" covered="1"/>
    <du var="chars" def="60" use="62" target="90" covered="1"/>
    <du var="chars" def="60" use="63" covered="1"/>
    <du var="chars" def="60" use="62" covered="1"/>
    <du var="chars" def="60" use="77" target="77" covered="1"/>
    <du var="chars" def="60" use="77" target="62" covered="1"/>
    <du var="chars" def="60" use="77" target="77" covered="1"/>
    <du var="chars" def="60" use="77" target="78" covered="1"/>
    <du var="chars" def="60" use="77" target="78" covered="1"/>
    <du var="chars" def="60" use="77" target="62" covered="1"/>
    <du var="chars" def="60" use="72" target="72" covered="1"/>
    <du var="chars" def="60" use="72" target="62" covered="0"/>
    <du var="chars" def="60" use="72" target="72" covered="0"/>
    <du var="chars" def="60" use="72" target="73" covered="1"/>
    <du var="chars" def="60" use="72" target="73" covered="0"/>
    <du var="chars" def="60" use="72" target="62" covered="0"/>
    <du var="i" def="62" use="62" target="63" covered="1"/>
    <du var="i" def="62" use="62" target="90" covered="0"/>

```


Figura 6.

Pit Test Coverage Report

Package Summary

org.apache.zookeeper.common

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
1	72%	80%	96%

Breakdown by Class

Name	Line Coverage	Mutation Coverage	Test Strength
PathUtils.java	72%	80%	96%

```

1. negated conditional → KILLED
2. negated conditional → KILLED
3. negated conditional → KILLED
4. negated conditional → KILLED
5. changed conditional boundary → SURVIVED
6. changed conditional boundary → SURVIVED
7. changed conditional boundary → KILLED
8. changed conditional boundary → KILLED
9. changed conditional boundary → KILLED
10. changed conditional boundary → KILLED
11. changed conditional boundary → KILLED
12. changed conditional boundary → KILLED
13. negated conditional → KILLED
14. negated conditional → KILLED
15. negated conditional → KILLED
16. negated conditional → KILLED

```

81

Figura 7.

```

▼<method name="createNode" desc="(Ljava/lang/String;[BLjava/util/List;)V">
<du var="this" def="445" use="465" covered="1"/>
<du var="this" def="445" use="472" covered="1"/>
<du var="this" def="445" use="488" covered="1"/>
<du var="this" def="445" use="489" covered="1"/>
<du var="this" def="445" use="490" covered="1"/>
<du var="this" def="445" use="523" covered="1"/>
<du var="this" def="445" use="529" covered="1"/>
<du var="this" def="445" use="530" covered="1"/>
<du var="this" def="445" use="531" covered="1"/>
<du var="this" def="445" use="527" covered="0"/>
<du var="this" def="445" use="519" covered="0"/>
<du var="this" def="445" use="516" covered="0"/>
<du var="this" def="445" use="497" covered="1"/>
<du var="this" def="445" use="500" covered="1"/>
<du var="this" def="445" use="495" covered="0"/>
<du var="this" def="445" use="493" covered="0"/>
<du var="path" def="445" use="489" covered="1"/>
<du var="path" def="445" use="490" covered="1"/>
<du var="path" def="445" use="523" covered="1"/>
<du var="path" def="445" use="529" covered="1"/>
<du var="path" def="445" use="530" covered="1"/>
<du var="path" def="445" use="503" covered="1"/>
<du var="path" def="445" use="495" covered="0"/>
<du var="path" def="445" use="493" covered="0"/>
<du var="data" def="445" use="486" covered="1"/>
<du var="data" def="445" use="524" target="524" covered="1"/>
<du var="data" def="445" use="524" target="524" covered="1"/>
<du var="data" def="445" use="524" covered="1"/>
<du var="acl" def="445" use="465" covered="1"/>
<du var="ephemeralOwner" def="445" use="491" covered="1"/>
<du var="ephemeralOwner" def="445" use="496" target="497" covered="1"/>
<du var="ephemeralOwner" def="445" use="496" target="506" covered="1"/>
<du var="ephemeralOwner" def="445" use="497" covered="1"/>
<du var="ephemeralOwner" def="445" use="500" covered="1"/>
<du var="parentCVersion" def="445" use="473" target="474" covered="0"/>
<du var="parentCVersion" def="445" use="473" target="482" covered="1"/>
<du var="parentCVersion" def="445" use="482" target="483" covered="1"/>
<du var="parentCVersion" def="445" use="482" target="486" covered="1"/>
<du var="parentCVersion" def="445" use="483" covered="1"/>
<du var="zxid" def="445" use="484" covered="1"/>

```

```

<du var="outputStat" def="445" use="506" target="507" covered="0"/>
<du var="outputStat" def="445" use="506" target="509" covered="1"/>
<du var="outputStat" def="445" use="507" covered="0"/>
<du var="this.nodes" def="445" use="472" covered="1"/>
<du var="this.nodes" def="445" use="488" covered="1"/>
<du var="this.nodes" def="445" use="490" covered="1"/>
<du var="this.aclCache" def="445" use="465" covered="1"/>
<du var="this.nodeDataSize" def="445" use="489" covered="1"/>
<du var="CONTAINER" def="445" use="492" target="493" covered="0"/>
<du var="CONTAINER" def="445" use="492" target="494" covered="1"/>
<du var="this.containers" def="445" use="493" covered="0"/>
<du var="TTL" def="445" use="494" target="495" covered="0"/>
<du var="TTL" def="445" use="494" target="496" covered="1"/>
<du var="this.ttls" def="445" use="495" covered="0"/>
<du var="this.ephemerals" def="445" use="497" covered="1"/>
<du var="this.ephemerals" def="445" use="500" covered="1"/>
<du var="this.pTrie" def="445" use="516" covered="0"/>
<du var="this.dataWatches" def="445" use="530" covered="1"/>
<du var="NodeCreated" def="445" use="530" covered="1"/>
<du var="this.childWatches" def="445" use="531" covered="1"/>
<du var="NodeChildrenChanged" def="445" use="531" covered="1"/>
<du var="parentName" def="446" use="472" covered="1"/>
<du var="parentName" def="446" use="488" covered="1"/>
<du var="parentName" def="446" use="511" target="513" covered="0"/>
<du var="parentName" def="446" use="511" target="523" covered="1"/>
<du var="parentName" def="446" use="531" target="531" covered="1"/>
<du var="parentName" def="446" use="531" target="531" covered="1"/>
<du var="parentName" def="446" use="531" covered="1"/>
<du var="parentName" def="446" use="519" covered="0"/>
<du var="parentName" def="446" use="516" covered="0"/>
<du var="childName" def="447" use="468" target="469" covered="0"/>
<du var="childName" def="447" use="468" target="472" covered="1"/>
<du var="childName" def="447" use="487" covered="1"/>
<du var="childName" def="447" use="513" target="516" covered="0"/>
<du var="childName" def="447" use="513" target="518" covered="0"/>
<du var="childName" def="447" use="518" target="519" covered="0"/>
<du var="childName" def="447" use="518" target="523" covered="0"/>
<du var="stat" def="448" use="486" covered="1"/>
<du var="parent" def="449" use="450" target="451" covered="1"/>
<du var="parent" def="449" use="450" target="453" covered="1"/>
<du var="parent" def="449" use="453" covered="1"/>
<du var="parent" def="449" use="453" covered="1"/>
<du var="parent" def="449" use="467" covered="1"/>
<du var="parent" def="449" use="472" covered="1"/>
<du var="parent" def="449" use="482" target="483" covered="1"/>
<du var="parent" def="449" use="482" target="486" covered="1"/>
<du var="parent" def="449" use="487" covered="1"/>
<du var="parent" def="449" use="488" covered="1"/>
<du var="parent" def="449" use="483" covered="1"/>
<du var="parent" def="449" use="484" covered="1"/>
<du var="parent" def="449" use="474" covered="0"/>
<du var="parent.stat" def="449" use="482" target="483" covered="1"/>
<du var="parent.stat" def="449" use="482" target="486" covered="1"/>
<du var="parent.stat" def="449" use="483" covered="1"/>
<du var="parent.stat" def="449" use="484" covered="1"/>
<du var="parent.stat" def="449" use="474" covered="0"/>
<du var="longval" def="465" use="486" covered="1"/>
<du var="children" def="467" use="468" target="469" covered="0"/>
<du var="children" def="467" use="468" target="472" covered="1"/>
<du var="parentCVersion" def="475" use="482" target="483" covered="0"/>
<du var="parentCVersion" def="475" use="482" target="486" covered="0"/>
<du var="parentCVersion" def="475" use="483" covered="0"/>
<du var="child" def="486" use="507" covered="0"/>
<du var="ephemeralType" def="491" use="492" target="493" covered="0"/>
<du var="ephemeralType" def="491" use="492" target="494" covered="1"/>
<du var="ephemeralType" def="491" use="494" target="495" covered="0"/>
<du var="ephemeralType" def="491" use="494" target="496" covered="1"/>
<du var="list" def="497" use="498" target="499" covered="1"/>
<du var="list" def="497" use="498" target="502" covered="0"/>
<du var="list" def="497" use="502" covered="0"/>
<du var="list" def="497" use="503" covered="0"/>
<du var="list" def="499" use="502" covered="1"/>
<du var="list" def="499" use="502" covered="1"/>
<du var="list" def="499" use="503" covered="1"/>
<du var="lastPrefix" def="523" use="526" target="527" covered="0"/>
<du var="lastPrefix" def="523" use="526" target="529" covered="1"/>
<du var="lastPrefix" def="523" use="527" covered="0"/>
<du var="bytes" def="524" use="529" covered="1"/>
<du var="bytes" def="524" use="527" covered="0"/>
<counter type="DU" missed="39" covered="82"/>
<counter type="METHOD" missed="0" covered="1"/>
</method>
```

```
▼<method name="deleteNode" desc="(Ljava/lang/String;)V">
<du var="this" def="544" use="556" covered="1"/>
<du var="this" def="544" use="564" covered="1"/>
<du var="this" def="544" use="567" covered="1"/>
<du var="this" def="544" use="571" covered="1"/>
<du var="this" def="544" use="573" covered="1"/>
<du var="this" def="544" use="574" covered="1"/>
<du var="this" def="544" use="604" covered="1"/>
<du var="this" def="544" use="614" covered="1"/>
<du var="this" def="544" use="627" covered="1"/>
<du var="this" def="544" use="628" covered="1"/>
<du var="this" def="544" use="629" covered="1"/>
<du var="this" def="544" use="611" covered="0"/>
<du var="this" def="544" use="600" covered="0"/>
<du var="this" def="544" use="588" covered="0"/>
<du var="this" def="544" use="586" covered="0"/>
<du var="this" def="544" use="584" covered="0"/>
<du var="path" def="544" use="567" covered="1"/>
<du var="path" def="544" use="571" covered="1"/>
<du var="path" def="544" use="574" covered="1"/>
<du var="path" def="544" use="604" covered="1"/>
<du var="path" def="544" use="614" covered="1"/>
<du var="path" def="544" use="627" covered="1"/>
<du var="path" def="544" use="628" covered="1"/>
<du var="path" def="544" use="617" covered="0"/>
<du var="path" def="544" use="591" covered="0"/>
<du var="path" def="544" use="586" covered="0"/>
<du var="path" def="544" use="584" covered="0"/>
<du var="zxid" def="544" use="561" target="562" covered="1"/>
<du var="zxid" def="544" use="561" target="564" covered="1"/>
<du var="zxid" def="544" use="562" covered="1"/>
<du var="this.nodes" def="544" use="556" covered="1"/>
<du var="this.nodes" def="544" use="564" covered="1"/>
<du var="this.nodes" def="544" use="567" covered="1"/>
<du var="this.nodes" def="544" use="571" covered="1"/>
<du var="this.aclCache" def="544" use="573" covered="1"/>
<du var="this.nodeDataSize" def="544" use="574" covered="1"/>
<du var="CONTAINER" def="544" use="583" target="584" covered="0"/>
<du var="CONTAINER" def="544" use="583" target="585" covered="1"/>
<du var="this.containers" def="544" use="584" covered="0"/>
<du var="TTL" def="544" use="585" target="586" covered="0"/>
<du var="TTL" def="544" use="585" target="587" covered="1"/>
<du var="this.ttls" def="544" use="586" covered="0"/>
<du var="this.ephemerals" def="544" use="588" covered="0"/>
<du var="this.pTrie" def="544" use="600" covered="0"/>
<du var="LOG" def="544" use="616" target="617" covered="0"/>
<du var="LOG" def="544" use="616" target="627" covered="1"/>
<du var="LOG" def="544" use="617" covered="0"/>
<du var="LOG" def="544" use="621" covered="0"/>
<du var="this.dataWatches" def="544" use="627" covered="1"/>
<du var="NodeDeleted" def="544" use="627" covered="1"/>
<du var="NodeDeleted" def="544" use="628" covered="1"/>
<du var="this.childWatches" def="544" use="628" covered="1"/>
<du var="this.childWatches" def="544" use="629" covered="1"/>
<du var="NodeChildrenChanged" def="544" use="629" covered="1"/>
<du var="parentName" def="545" use="556" covered="1"/>
<du var="parentName" def="545" use="564" covered="1"/>
<du var="parentName" def="545" use="597" target="597" covered="0"/>
<du var="parentName" def="545" use="597" target="604" covered="1"/>
<du var="parentName" def="545" use="629" target="629" covered="1"/>
<du var="parentName" def="545" use="629" target="629" covered="1"/>
<du var="parentName" def="545" use="629" covered="1"/>
<du var="parentName" def="545" use="621" covered="0"/>
<du var="parentName" def="545" use="600" covered="0"/>
<du var="childName" def="546" use="557" covered="1"/>
<du var="childName" def="546" use="597" target="600" covered="0"/>
<du var="childName" def="546" use="597" target="604" covered="0"/>
<du var="parent" def="551" use="552" target="553" covered="0"/>
<du var="parent" def="551" use="552" target="555" covered="1"/>
<du var="parent" def="551" use="555" covered="1"/>
<du var="parent" def="551" use="556" covered="1"/>
<du var="parent" def="551" use="557" covered="1"/>
<du var="parent" def="551" use="561" target="562" covered="1"/>
<du var="parent" def="551" use="561" target="564" covered="1"/>
<du var="parent" def="551" use="564" covered="1"/>
<du var="parent" def="551" use="580" covered="1"/>
<du var="parent" def="551" use="580" covered="1"/>
<du var="parent" def="551" use="562" covered="1"/>
<du var="parent.stat" def="551" use="561" target="562" covered="1"/>
<du var="parent.stat" def="551" use="561" target="564" covered="1"/>
<du var="parent.stat" def="551" use="562" covered="1"/>
<du var="node" def="567" use="568" target="569" covered="1"/>
<du var="node" def="567" use="568" target="571" covered="1"/>
<du var="node" def="567" use="572" covered="1"/>
<du var="node" def="567" use="572" covered="1"/>
<du var="node" def="567" use="573" covered="1"/>
<du var="node" def="567" use="574" covered="1"/>
<du var="node" def="567" use="581" covered="1"/>
<du var="node" def="567" use="608" covered="0"/>
<du var="node" def="567" use="608" covered="0"/>
<du var="node" def="567" use="609" target="609" covered="0"/>
<du var="node" def="567" use="609" target="609" covered="0"/>
```

```

<du var="node" def="567" use="609" covered="0"/>
<du var="node.acl" def="567" use="573" covered="1"/>
<du var="node.data" def="567" use="574" covered="1"/>
<du var="node.data" def="567" use="609" target="609" covered="0"/>
<du var="node.data" def="567" use="609" target="609" covered="0"/>
<du var="node.data" def="567" use="609" covered="0"/>
<du var="node.stat" def="567" use="581" covered="1"/>
<du var="eowner" def="581" use="587" target="588" covered="0"/>
<du var="eowner" def="581" use="587" target="595" covered="1"/>
<du var="eowner" def="581" use="588" covered="0"/>
<du var="ephemeralType" def="582" use="583" target="584" covered="0"/>
<du var="ephemeralType" def="582" use="583" target="585" covered="1"/>
<du var="ephemeralType" def="582" use="585" target="586" covered="0"/>
<du var="ephemeralType" def="582" use="587" target="587" covered="1"/>
<du var="nodes" def="588" use="589" target="590" covered="0"/>
<du var="nodes" def="588" use="589" target="595" covered="0"/>
<du var="nodes" def="588" use="590" covered="0"/>
<du var="nodes" def="588" use="590" covered="0"/>
<du var="nodes" def="588" use="591" covered="0"/>
<du var="lastPrefix" def="604" use="605" target="607" covered="0"/>
<du var="lastPrefix" def="604" use="605" target="614" covered="1"/>
<du var="lastPrefix" def="604" use="611" covered="0"/>
<counter type="DU" missed="44" covered="73"/>
<counter type="METHOD" missed="0" covered="1"/>
</method>

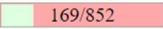
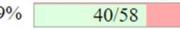
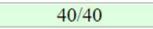
```

Figura 8.

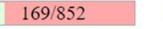
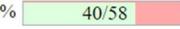
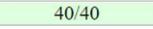
Pit Test Coverage Report

Package Summary

`org.apache.zookeeper.server`

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
1	20%  169/852	69%  40/58	100%  40/40

Breakdown by Class

Name	Line Coverage	Mutation Coverage	Test Strength
<code>DataTree.java</code>	20%  169/852	69%  40/58	100%  40/40

529	1. removed call to <code>org/apache/zookeeper/server/DataTree::updateWriteStat</code> → <code>TIMED_OUT</code>
531	1. negated conditional → <code>TIMED_OUT</code>
546	1. Replaced integer addition with subtraction → <code>TIMED_OUT</code>
552	1. negated conditional → <code>TIMED_OUT</code>
556	1. removed call to <code>org/apache/zookeeper/server/NodeHashMap::preChange</code> → <code>TIMED_OUT</code>
561	1. changed conditional boundary → <code>TIMED_OUT</code>
562	2. negated conditional → <code>TIMED_OUT</code>
562	1. removed call to <code>org/apache/zookeeper/data/StatPersisted::setPzxid</code> → <code>TIMED_OUT</code>
564	1. removed call to <code>org/apache/zookeeper/server/NodeHashMap::postChange</code> → <code>TIMED_OUT</code>
568	1. negated conditional → <code>TIMED_OUT</code>
573	1. removed call to <code>org/apache/zookeeper/server/ReferenceCountedACLCache::removeUsage</code> → <code>TIMED_OUT</code>
574	1. removed negation → <code>TIMED_OUT</code>
583	1. negated conditional → <code>TIMED_OUT</code>
585	1. negated conditional → <code>TIMED_OUT</code>

Figura 9.

```

▼<method name="createNode" desc="(Ljava/lang/String;
[BLjava/util/List;Ljava/org/apache/zookeeper/data/Stat;)V">
<du var="this" def="445" use="465" covered="1"/>
<du var="this" def="445" use="472" covered="1"/>
<du var="this" def="445" use="488" covered="1"/>
<du var="this" def="445" use="489" covered="1"/>
<du var="this" def="445" use="490" covered="1"/>
<du var="this" def="445" use="523" covered="1"/>
<du var="this" def="445" use="529" covered="1"/>
<du var="this" def="445" use="530" covered="1"/>
<du var="this" def="445" use="531" covered="1"/>
<du var="this" def="445" use="527" covered="0"/>
<du var="this" def="445" use="519" covered="0"/>
<du var="this" def="445" use="516" covered="0"/>
<du var="this" def="445" use="497" covered="1"/>
<du var="this" def="445" use="500" covered="1"/>
<du var="this" def="445" use="495" covered="1"/>
<du var="this" def="445" use="493" covered="1"/>
<du var="path" def="445" use="489" covered="1"/>
<du var="path" def="445" use="490" covered="1"/>
<du var="path" def="445" use="523" covered="1"/>
<du var="path" def="445" use="529" covered="1"/>
<du var="path" def="445" use="530" covered="1"/>
<du var="path" def="445" use="503" covered="1"/>
<du var="path" def="445" use="495" covered="1"/>
<du var="path" def="445" use="493" covered="1"/>
<du var="data" def="445" use="486" covered="1"/>
<du var="data" def="445" use="524" target="524" covered="1"/>
<du var="data" def="445" use="524" target="524" covered="1"/>
<du var="data" def="445" use="524" covered="1"/>
<du var="acl" def="445" use="465" covered="1"/>
<du var="ephemeralOwner" def="445" use="491" covered="1"/>
<du var="ephemeralOwner" def="445" use="496" target="497" covered="1"/>
<du var="ephemeralOwner" def="445" use="496" target="506" covered="1"/>
<du var="ephemeralOwner" def="445" use="497" covered="1"/>
<du var="ephemeralOwner" def="445" use="500" covered="1"/>
<du var="parentCVersion" def="445" use="473" target="474" covered="0"/>
<du var="parentCVersion" def="445" use="473" target="482" covered="1"/>
<du var="parentCVersion" def="445" use="482" target="483" covered="1"/>
<du var="parentCVersion" def="445" use="482" target="486" covered="1"/>
<du var="parentCVersion" def="445" use="483" covered="1"/>
<du var="xid" def="445" use="484" covered="1"/>
<du var="outputStat" def="445" use="506" target="507" covered="0"/>
<du var="outputStat" def="445" use="506" target="509" covered="1"/>
<du var="outputStat" def="445" use="507" covered="0"/>
<du var="this.nodes" def="445" use="472" covered="1"/>
<du var="this.nodes" def="445" use="488" covered="1"/>
<du var="this.nodes" def="445" use="490" covered="1"/>
<du var="this.aclCache" def="445" use="465" covered="1"/>
<du var="this.nodeDataSize" def="445" use="489" covered="1"/>
<du var="CONTAINER" def="445" use="492" target="493" covered="1"/>
<du var="CONTAINER" def="445" use="492" target="494" covered="1"/>
<du var="this.containers" def="445" use="493" covered="1"/>
<du var="TTL" def="445" use="494" target="495" covered="1"/>
<du var="TTL" def="445" use="494" target="496" covered="1"/>
<du var="this.ttls" def="445" use="495" covered="1"/>
<du var="this.ephemerals" def="445" use="497" covered="1"/>
<du var="this.ephemerals" def="445" use="500" covered="1"/>
<du var="this.pTrie" def="445" use="516" covered="0"/>
<du var="this.dataWatches" def="445" use="530" covered="1"/>
<du var="NodeCreated" def="445" use="530" covered="1"/>
<du var="this.childWatches" def="445" use="531" covered="1"/>
<du var="NodeChildrenChanged" def="445" use="531" covered="1"/>
<du var="parentName" def="446" use="472" covered="1"/>
<du var="parentName" def="446" use="488" covered="1"/>
<du var="parentName" def="446" use="511" target="513" covered="0"/>
<du var="parentName" def="446" use="511" target="523" covered="1"/>
<du var="parentName" def="446" use="531" target="531" covered="1"/>
<du var="parentName" def="446" use="531" target="531" covered="1"/>
<du var="parentName" def="446" use="531" covered="1"/>
<du var="parentName" def="446" use="519" covered="0"/>
<du var="parentName" def="446" use="516" covered="0"/>
<du var="childName" def="447" use="468" target="469" covered="1"/>
<du var="childName" def="447" use="468" target="472" covered="1"/>
<du var="childName" def="447" use="487" covered="1"/>
<du var="childName" def="447" use="513" target="516" covered="0"/>
<du var="childName" def="447" use="513" target="518" covered="0"/>
<du var="childName" def="447" use="518" target="519" covered="0"/>
<du var="childName" def="447" use="518" target="523" covered="0"/>
<du var="stat" def="448" use="486" covered="1"/>
<du var="parent" def="449" use="450" target="451" covered="1"/>
```

```

<du var="parent" def="449" use="450" target="453" covered="1"/>
<du var="parent" def="449" use="453" covered="1"/>
<du var="parent" def="449" use="453" covered="1"/>
<du var="parent" def="449" use="467" covered="1"/>
<du var="parent" def="449" use="472" covered="1"/>
<du var="parent" def="449" use="482" target="483" covered="1"/>
<du var="parent" def="449" use="482" target="486" covered="1"/>
<du var="parent" def="449" use="487" covered="1"/>
<du var="parent" def="449" use="488" covered="1"/>
<du var="parent" def="449" use="483" covered="1"/>
<du var="parent" def="449" use="484" covered="1"/>
<du var="parent" def="449" use="474" covered="0"/>
<du var="parent.stat" def="449" use="482" target="483" covered="1"/>
<du var="parent.stat" def="449" use="482" target="486" covered="1"/>
<du var="parent.stat" def="449" use="483" covered="1"/>
<du var="parent.stat" def="449" use="484" covered="1"/>
<du var="parent.stat" def="449" use="474" covered="0"/>
<du var="longval" def="465" use="486" covered="1"/>
<du var="children" def="467" use="468" target="469" covered="1"/>
<du var="children" def="467" use="468" target="472" covered="1"/>
<du var="parentCVersion" def="475" use="482" target="483" covered="0"/>
<du var="parentCVersion" def="475" use="482" target="486" covered="0"/>
<du var="parentCVersion" def="475" use="483" covered="0"/>
<du var="child" def="488" use="507" covered="0"/>
<du var="ephemeralType" def="491" use="492" target="493" covered="1"/>
<du var="ephemeralType" def="491" use="492" target="494" covered="1"/>
<du var="ephemeralType" def="491" use="494" target="495" covered="1"/>
<du var="ephemeralType" def="491" use="494" target="496" covered="1"/>
<du var="list" def="497" use="498" target="499" covered="1"/>
<du var="list" def="497" use="498" target="502" covered="0"/>
<du var="list" def="497" use="502" covered="0"/>
<du var="list" def="497" use="502" covered="0"/>
<du var="list" def="499" use="502" covered="1"/>
<du var="list" def="499" use="502" covered="1"/>
<du var="list" def="499" use="503" covered="1"/>
<du var="lastPrefix" def="523" use="526" target="527" covered="0"/>
<du var="lastPrefix" def="523" use="526" target="529" covered="1"/>
<du var="lastPrefix" def="523" use="527" covered="0"/>
<du var="bytes" def="524" use="529" covered="1"/>
<du var="bytes" def="524" use="527" covered="0"/>
<counter type="DU" missed="27" covered="94"/>
<counter type="METHOD" missed="0" covered="1"/>
</method>
▼<method name="deleteNode" desc="(Ljava/lang/String;)V">

▼<method name="deleteNode" desc="(Ljava/lang/String;)V">
<du var="this" def="544" use="556" covered="1"/>
<du var="this" def="544" use="564" covered="1"/>
<du var="this" def="544" use="567" covered="1"/>
<du var="this" def="544" use="571" covered="1"/>
<du var="this" def="544" use="573" covered="1"/>
<du var="this" def="544" use="574" covered="1"/>
<du var="this" def="544" use="604" covered="1"/>
<du var="this" def="544" use="614" covered="1"/>
<du var="this" def="544" use="627" covered="1"/>
<du var="this" def="544" use="628" covered="1"/>
<du var="this" def="544" use="629" covered="1"/>
<du var="this" def="544" use="611" covered="0"/>
<du var="this" def="544" use="600" covered="0"/>
<du var="this" def="544" use="588" covered="1"/>
<du var="this" def="544" use="586" covered="1"/>
<du var="this" def="544" use="584" covered="1"/>
<du var="path" def="544" use="567" covered="1"/>
<du var="path" def="544" use="571" covered="1"/>
<du var="path" def="544" use="574" covered="1"/>
<du var="path" def="544" use="604" covered="1"/>
<du var="path" def="544" use="614" covered="1"/>
<du var="path" def="544" use="627" covered="1"/>
<du var="path" def="544" use="628" covered="1"/>
<du var="path" def="544" use="617" covered="0"/>
<du var="path" def="544" use="591" covered="1"/>
<du var="path" def="544" use="586" covered="1"/>
<du var="path" def="544" use="584" covered="1"/>
<du var="zxid" def="544" use="561" target="562" covered="1"/>
<du var="zxid" def="544" use="561" target="564" covered="1"/>
<du var="zxid" def="544" use="562" covered="1"/>
<du var="this.nodes" def="544" use="556" covered="1"/>
<du var="this.nodes" def="544" use="564" covered="1"/>
<du var="this.nodes" def="544" use="567" covered="1"/>
<du var="this.nodes" def="544" use="571" covered="1"/>
<du var="this.aclCache" def="544" use="573" covered="1"/>
<du var="this.nodeDataSize" def="544" use="574" covered="1"/>
<du var="CONTAINER" def="544" use="583" target="584" covered="1"/>
<du var="CONTAINER" def="544" use="583" target="585" covered="1"/>
<du var="this.containers" def="544" use="584" covered="1"/>
<du var="TTL" def="544" use="585" target="586" covered="1"/>
<du var="TTL" def="544" use="585" target="587" covered="1"/>
<du var="this.ttls" def="544" use="586" covered="1"/>
<du var="this.ephemerals" def="544" use="588" covered="1"/>
<du var="this.pTrie" def="544" use="600" covered="0"/>
<du var="LOG" def="544" use="616" target="617" covered="0"/>
<du var="LOG" def="544" use="616" target="627" covered="1"/>

```

```
<du var="LOG" def="544" use="617" covered="0"/>
<du var="LOG" def="544" use="621" covered="0"/>
<du var="this.dataWatches" def="544" use="627" covered="1"/>
<du var="NodeDeleted" def="544" use="628" covered="1"/>
<du var="this.childWatches" def="544" use="628" covered="1"/>
<du var="this.childWatches" def="544" use="629" covered="1"/>
<du var="NodeChildrenChanged" def="544" use="629" covered="1"/>
<du var="parentName" def="545" use="556" covered="0"/>
<du var="parentName" def="545" use="564" covered="1"/>
<du var="parentName" def="545" use="597" target="597" covered="0"/>
<du var="parentName" def="545" use="597" target="604" covered="1"/>
<du var="parentName" def="545" use="629" target="629" covered="1"/>
<du var="parentName" def="545" use="629" target="629" covered="1"/>
<du var="parentName" def="545" use="629" covered="1"/>
<du var="parentName" def="545" use="621" covered="0"/>
<du var="parentName" def="545" use="600" covered="0"/>
<du var="childName" def="546" use="557" covered="1"/>
<du var="childName" def="546" use="597" target="600" covered="0"/>
<du var="childName" def="546" use="597" target="604" covered="0"/>
<du var="parent" def="551" use="552" target="553" covered="1"/>
<du var="parent" def="551" use="552" target="555" covered="1"/>
<du var="parent" def="551" use="555" covered="1"/>
<du var="parent" def="551" use="555" covered="1"/>
<du var="parent" def="551" use="556" covered="1"/>
<du var="parent" def="551" use="557" covered="1"/>
<du var="parent" def="551" use="561" target="562" covered="1"/>
<du var="parent" def="551" use="561" target="564" covered="1"/>
<du var="parent" def="551" use="564" covered="1"/>
<du var="parent" def="551" use="580" covered="1"/>
<du var="parent" def="551" use="580" covered="1"/>
<du var="parent" def="551" use="562" covered="1"/>
<du var="parent.stat" def="551" use="561" target="562" covered="1"/>
<du var="parent.stat" def="551" use="561" target="564" covered="1"/>
<du var="parent.stat" def="551" use="562" covered="1"/>
<du var="node" def="567" use="568" target="569" covered="1"/>
<du var="node" def="567" use="568" target="571" covered="1"/>
<du var="node" def="567" use="572" covered="1"/>
<du var="node" def="567" use="572" covered="1"/>
<du var="node" def="567" use="573" covered="1"/>
<du var="node" def="567" use="574" covered="1"/>
<du var="node" def="567" use="581" covered="1"/>
<du var="node" def="567" use="608" covered="0"/>
<du var="node" def="567" use="608" covered="0"/>
<du var="node" def="567" use="609" target="609" covered="0"/>
<du var="node" def="567" use="609" target="609" covered="0"/>
<du var="node" def="567" use="609" covered="0"/>
<du var="node.acl" def="567" use="573" covered="1"/>
<du var="node.data" def="567" use="574" covered="1"/>
<du var="node.data" def="567" use="609" target="609" covered="0"/>
<du var="node.data" def="567" use="609" target="609" covered="0"/>
<du var="node.data" def="567" use="609" covered="0"/>
<du var="node.stat" def="567" use="581" covered="1"/>
<du var="eowner" def="581" use="587" target="588" covered="1"/>
<du var="eowner" def="581" use="587" target="595" covered="1"/>
<du var="eowner" def="581" use="588" covered="1"/>
<du var="ephemeralType" def="582" use="583" target="584" covered="1"/>
<du var="ephemeralType" def="582" use="583" target="585" covered="1"/>
<du var="ephemeralType" def="582" use="585" target="586" covered="1"/>
<du var="ephemeralType" def="582" use="585" target="587" covered="1"/>
<du var="nodes" def="588" use="589" target="590" covered="1"/>
<du var="nodes" def="588" use="589" target="595" covered="0"/>
<du var="nodes" def="588" use="590" covered="1"/>
<du var="nodes" def="588" use="591" covered="1"/>
<du var="lastPrefix" def="604" use="605" target="607" covered="0"/>
<du var="lastPrefix" def="604" use="605" target="614" covered="1"/>
<du var="lastPrefix" def="604" use="611" covered="0"/>
<counter type="DU" missed="24" covered="93"/>
<counter type="METHOD" missed="0" covered="1"/>
</method>
```

Figura 10.

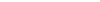
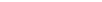
Pit Test Coverage Report

Package Summary

org.apache.zookeeper.server

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
1	21%	181/852	33% 19/58 46% 19/41

Breakdown by Class

Name	Line Coverage	Mutation Coverage	Test Strength
DataTree.java	21%  181/852	33%  19/58	46%  19/41

Mutations

```

301 1. removed call to org/apache/zookeeper/server/DataTree::addConfigNode → KILLED
303 1. removed call to java/util/concurrent/atomic/AtomicLong::set → SURVIVED
309 1. removed call to org/apache/zookeeper/util/ServiceUtils::requestSystemExit → NO_COVERAGE
422 1. removed call to org/apache/zookeeper/server/DataTree::createNode → KILLED
447 1. Replaced integer addition with subtraction → KILLED
450 1. negated conditional → KILLED
468 1. negated conditional → KILLED
472 1. removed call to org/apache/zookeeper/server/NodeHashMap::preChange → SURVIVED
473 1. negated conditional → KILLED
475 1. Changed increment from 1 to -1 → NO_COVERAGE
482 1. negated conditional → KILLED
2. changed conditional boundary → SURVIVED
483 1. removed call to org/apache/zookeeper/data/StatPersisted::setCversion → KILLED
484 1. removed call to org/apache/zookeeper/data/StatPersisted::setPzxid → SURVIVED
488 1. removed call to org/apache/zookeeper/server/NodeHashMap::postChange → SURVIVED
492 1. negated conditional → KILLED
494 1. negated conditional → KILLED
496 1. negated conditional → KILLED
498 1. negated conditional → KILLED
506 1. negated conditional → KILLED
507 1. removed call to org/apache/zookeeper/server/DataNode::copyStat → NO_COVERAGE
511 1. negated conditional → SURVIVED
513 1. negated conditional → NO_COVERAGE
516 1. removed call to org/apache/zookeeper/common/PathTrie::addPath → NO_COVERAGE
518 1. negated conditional → NO_COVERAGE
519 1. removed call to org/apache/zookeeper/server/DataTree::updateQuotaForPath → NO_COVERAGE
524 1. negated conditional → KILLED
526 1. negated conditional → SURVIVED
527 1. removed call to org/apache/zookeeper/server/DataTree::updateQuotaStat → NO_COVERAGE
529 1. removed call to org/apache/zookeeper/server/DataTree::updateWriteStat → SURVIVED
531 1. negated conditional → SURVIVED
546 1. Replaced integer addition with subtraction → KILLED
552 1. negated conditional → KILLED
556 1. removed call to org/apache/zookeeper/server/NodeHashMap::preChange → SURVIVED
561 1. negated conditional → KILLED
2. changed conditional boundary → SURVIVED
562 1. removed call to org/apache/zookeeper/data/StatPersisted::setPzxid → KILLED
564 1. removed call to org/apache/zookeeper/server/NodeHashMap::postChange → SURVIVED
568 1. negated conditional → KILLED

```

Tabella 1.

PATH	DATA	ACL	E.O.	PCV	ZXID	TIME
“a”	[b1, ..., bn]	[]	0	0	0	0
“node1/node2”	[b1, ..., bn]	Null	0xFF000000000000001L	0	-1	1
“/.../node”	Null	OPEN_ACL_UNSAFE	0x8000000000000000L	0	1	3x10 ⁵
“/a”	[b1, ..., bn]	CREATOR_ALL_ACL	0	3	1	1
“/abc”	[b1, ..., bn]	OPEN_ACL_UNSAFE	122	0	1	1
“/abcd56”	[]	CREATOR_ALL_ACL	0	-2	1	1
“/a??9m”	Null	READ_ACL_UNSAFE	122	0	1	1
“/a/b”	[b1, ..., bn]	OPEN_ACL_UNSAFE	0	0	1	1
“/a/b”	[b1, ..., bn]	READ_ACL_UNSAFE	2	0	1	1
“/a/b”	[b1, ..., bn]	OPEN_ACL_UNSAFE	0	1	1X10 ⁴	1
“/a/b”	[b1, ..., bn]	READ_ACL_UNSAFE	23	1	1	1
“/node.1/node.2/no de.3”	[b1, ..., bn]	CREATOR_ALL_ACL	0xFF0000000000111L	0	0	1
“/n1/n2/n3/n4”	[b1, ..., bn]	READ_ACL_UNSAFE	0xFF0000000000111L	0	0	1
“/n1/n2/n3/n4/n5/n6 /n7/n8”	[b1, ..., bn]	READ_ACL_UNSAFE	0x8000000000000000L	0	0	1