Test sul funzionamento del server dopo gli interventi in materia di scritture e locking

Questi sono gli esiti dei test di carico fatti con SoapUI per verificare se l'intervento di revisione completa di accesso alle header dei files e di flushing dei files ha dato benefici o meno. Il test compiuto randomizza una serie di operazioni che vanno dalla selezione, acquisizione dei titoli dei documenti caricamento di un documento selezionato, eventuale caricamento per modifica, eventuale modifica o sblocco del documento. Inoltre è previsto un salvataggio di documenti del tutto nuovi cui viene fatto far seguito da un reload del documento e successiva modifica (per simulare uno degli usi più frequenti nell'applicazione xdocway). Le modifiche ai documenti ed i loro inserimenti vengono regolati in maniera di avere un comportamento casuale ma orientato a privilegiare uno dei due comportamenti. Nelle statistiche, però, le modifiche pure e semplici hanno un conteggio distinto da quelle che vengono compiute “d'ufficio” dopo l'inserimento.

Archivi usati per i test

I test sono stati effettuati usando i seguenti archivi:

1. test22base: Directory che contiene un archivio fatto per il server di classe 22 da usarsi nei test successivi
2. test22: Directory che contiene l'archivio fatto per il server 22 che viene poi sperimentato con la modalità xml oppure buddy
3. test23.base.xml e test23.base.buddy: Directory contenenti gli archivi per il server 23 da usarsi nei successivi test.
4. test23a e test23b: In queste due directory vengono effettiati i diversi test. La directory 'a' si riferisce ai test fatti con server di tipo 23 prima degli interventi natalizi (hotwip_20090803) mentre la directory 'b' si riferisce a test fatti con in server 23 con le modifiche natalizie (hotwip_20091225).
5. test23.baseixd.xml, test23.baseidx.buddy, test23c e test23d: come il caso precedente ma con un archivio di base creato per usare ancora l'IDX anziché il B+Tree. In questo caso la directory 'c' si riferisce al server 23 pre-natalizio e la directory 'd' a quello natalizio.

Inizio dei test: Crash!

L'inizio dei test è stato caratterizzato da crash del server 23 nella sua versione sia nataliza che pre-natalizia. Ciò avviene quando i server sono in batteria. Con betterie di 5 o 10 server in parallelo, tutti release, il crash si presenta molto rapidamente, quasi subito. Poi, in verità, dopo essersi presentato una prima volta, pare non presentarsi oltre (o per lo meno non per molto tempo ancora). Con batterie di 5 server in debug si presenta ma ci mette un po' a saltar fuori. Quando salta fuori si presenta quel che segue.

funzione findAdd() in BTPAlgorithms.h riga 700.

Si evocano le varie funzioni per identificare nodeRad. Ci si domanda se nodeRad sia una foglia, lo è. Poi ci si domanda il valore di 'ret' per capire se la parola sia già presente, non è presente. Ci si domanda se il nodo sia pieno, linea 737, e non lo è. Si procede con l'evocare la nodeRad→add indicando retIndex pari a 0x5c (???)

Entrando nella add (BPTNode.h, riga 48) si arriva alla riga 53 in cui si fa una memmove per far posto alla parola che si intende introdurre. La cosa che lascia un po' perplessi è il fatto che il membro elems_ è enumerato da 0x00 a 0x5B quindi l'indice 0x5C è oltre la sua fine. Al comtempo, però, il valore di size_ è pari a 0x04de. Ora... mi pare che il calcolo sia fatto in modo bislacco. L'indice 0x5C suona tanto di nodo pieno anche quando il nodo risulta non esserlo ancora ed al contempo il valore di size_ pari a 0x04de mi pare errato. Anzi, perfeziono l'affermazione: se size_ è coerente, allora il calcolo della parte da spostare è errato, in caso contrario se non è coerente il valore è stato sporcato. In ambo i casi c'è un problema evidente.

Ogni successivo test non potrà considerarsi del tutto signficativo non essendo possibile svolgerlo su una batteria di server

Proseguimento senza batteria

Tutti i test riportati in queste directory sono stati effettuati usando un solo thread con intervalli tra un comando e l'altro variabili tra i 15 ed i 30 millisecondi. Il test si protrae per 180 secondi. Quello che si nota, operando con un singolo thread, è che i valori che vengono dati, come minimo e massimo, si riferiscono a tutti i thread, come se ce ne fossero pià di uno, ed il valore minimo è in realtà il massimo dei minimi così come il valore massimo è il massimo dei massimi, almeno a quanto pare. Il che ci dice che questi valori sono rappresentativi, per eccesso, solo di fronte ad un elevato numero di threads, ma su un singolo thread non sono in realtà in grado di dare un tangibile valor medio. Comunque, i test sono stati effettuati, in ogni sessione, sia utilizzando il repository di default, sia quello sotto forma di buddy file binario non compresso per verificare quant'impatto esso abbia nella gestione complessiva.

Una prima sessione di test è stata effettuata con server debug 22, 23 della versione prima della revisione del trattamento dei files, etichettata come appartenente al branch hotwip_20090803 e 23 della versione natalizia, quella che si “fregia” d'aver rimesso mano alla gestione dei files riducendo letture e flush in gran quantità, identificata come appartenente al branch hotwip_20091225.

Una seconda sessione di test è stata effettuata con le stesse versioni di cui sopra ma nella loro forma release.

Una terza sessione di test è stata effettuata con le stesse versioni release ma innalzando il livello di logging.

Prime impressioni

Una prima, primissima impressione si ha verificando che il server di classe 22 riesce a compiere un numero di inserimenti e modifiche molto superiore a quello del server 23, quale che esso sia. Tale cifra è oltre il doppio di quella spuntata dal server 23. Tra i due server 23 la differenza è minima, non molto apprezzabile. Anche introducendo la variabile buddy/xml la differenza non è

particolarmente sensibile, a parità di server. Nello stesso tempo essi producono all'incirca le stesse operazioni con una differenza non apprezabile.

Quello che invece risulta evidente in una forma diversa è un tema non meno preoccupante. Il server di classe 23 guadagna un 20/25% di tempo in sede di salvataggio di un documento rispetto allo stesso server di classe 22 ma si perde, letteralmente, in cose ignote in fase di modifica mettendoci un tempo oltre 5 volte superiore!

La terza sessione di test dovrebbe consentirci di identificare i punti in cui si è ottenuto un peggioramento anziché un miglioramento. Esiste in fine un'ulteriore possibilità: rifare i test della classe 23 partendo da un archivio già esistente in classe 22 che faccia produrre dei file BTree al posto degli attuali B+Tree.

Ulteriori Test

Viene compiuta una quarta sessione di test, in tutto corrispondente alla terza e quindi aggiunta ad essa. In questa sessione di test si compiono gli stessi passi effettuati per i due server di classe 23, pre e post natalizi, ma con archivi che usino il BTree anziché il B+Tree, sempre in condizione di elevato livello di logging e per questo, come detto, associato allo stesso terzo gruppo di rilevamenti.

Seconde impressioni

Le seconde impressioni sono buone e cattive al contempo. Buone perché ci consentono di identificare con una certa convinzione e precisione il protagonista, in negativo, del nostro problema principale, cattive perché ci dicono che lo sforzo compiuto non ha portato affatto gli effetti desiderati, anzi, l'esatto opposto. Il peggioramento, se possibile, è di gran lunga superiore a qualsiasi miglioramento ottenuto/ottenibile. Se non altro il campo d'azione è ora circoscritto e si possono fare valutazioni più di dettaglio. In realtà ci viene detta anche un'altra cosa. Utilizzando in ambo i casi il BTree, anziché il B+Tree, e quindi appiattendo una parte delle differenze che intercorrono tra il server di classe 22 e quello di classe 23, si nota che il server di classe 23 ha una capacità di salvataggio documenti superiore, nella stessa unità di tempo. Se il server 22 si attesta tra gli 88 e gli 89 con e senza buddy, il server 23 post natalizio si attesta attorno agli 88 nella modalità buddy e sale a 93 in quella XML ottenendo risultati comunque superiori al 23 pre natalizio.

Un maggior livello di dettaglio

Per scendere maggiormente nel dettaglio ho preso due modifiche a caso, di fatto la prima modifica al primo documento dell'archivio. Ho preso a campione il server di tipo 23 nella modalità post natalizia e facente uso degli XML quindi in condizioni, di fatto, corrispondenti. L'unica differenza consta nell'uso del B+Tree (test23b-xml) e del BTree (test23d-xml). L'esecuzione passo passo ci dice che le differenze prima e dopo l'indicizzazione possono essere trascurabili (nell'ordine di millesimi o pochi centesimi di secondo) mentre è la funzione di indicizzazione quella che evidenzia i veri problemi. Nella fattispecie il tempo che intercorre dal momento in cui l'indicizzazione ha inizio a quando si procede allo scaricamento del Tree delle chiavi è simile e trascurabile così come il tempo che segue la fine di tale scaricamento e la fine globale dell'indicizzazione. Quello che si nota, con sgomento, è che pur scaricando un numero di chiavi tutto sommato molto simile (201 nel caso lento 'b', 207 nel caso veloce 'd') la modalità 'b' impiega circa 2.4“ mente la modalità 'd' impiega 0 millesimi! Urgono ulteriori test per stringere il cerchio attorno al nostro candidato principe, ovvero al B+Tree, ed identificarne le idiosincrasie.

Primo girone

Dopo il compare tra i precedenti risultati, un ulteriore test effettuato esclusivamente sul server 23 post natalizio ha dato evidenza che la funzione di inserimento delle chiavi nel B+Tree è quella che evidenzia la propria sofferenza. Per tale ragione si replica nuovamente il test, sempre con la valutazione di cicli e di tempi, per valutare quanto sia il tempo di accesso al vocabolario e quanto quello di acesso al file VCB.

Secondo girone

Un ulteriore ciclo di test corrispondente al precedente, ma focalizzato sull'inserimento della chiave, pone in evidenza come sia la bt_insert_key() la funzione incriminata che, pur eseguita lo stesso numero di volte, provoca un comportamento assolutamente non paragonabile. A questo punto si ritiene sia il caso di affidarsi a sessioni di debugging di dettaglio.

Terzo girone

A seguito di una breve sessione di debug è stato evidente che in fase di modifica di un documento il B+Tree non veniva sottoposto a cache e questo ne deteriorava immensamente le prestazioni. Ora siamo del tutto in linea con il comportamento del BTree ed anzi, sottoponendolo a cache a sua volta, la differenza è minima se pure si rileva ancora un minimo vantaggio nei confronti del BTree. C' per altro credibile che tale vantaggio svanisca al crescere delle dimensioni dell'archivio sul quale si opera per la natura stessa del B+Tree.

Quello che si dovrebbe fare ora si divide in due aspetti:

1. Assicurarsi che il BTree/B+Tree in forma cached garantisca la correttezza del risultato anche in sede di indicizzazione differenziale e non solo in caso di indicizzazione incrementale.
2. Tentare l'applicazione della cache in indicizzazione incrementale per valutare il server 22 che risultati è in grado di offrire.

Sarebbe inoltre utile studiare un diverso sitema di logging che consenta di fare time test anche al volo, senza dover intervenire sul codice, ma la sua generalizzazione potrebbe risultare svantaggiosa rispetto al sistema corrente.

In quanto all'assicurazione che il btree sia consistente, non ci sono ragioni di dubitarne visto il metodo con il quale esso viene

gestito in questo frangente. Se opera correttamente in indicizzazione incrementale dovrebbe fare altrettanto in caso differenziale, tanto più che si tratta esclusivamente di operazioni di inserimento di chiavi nel btree. Rimane la necessità di apportare al server 22 la stessa correzione per verificare come si comporta in un simile scenario.

Quarto girone

Effettuando il test anche con una versione modificata del server di tipo 22 la differenza non si percepisce. Spieghiamo meglio il concetto. Mente nel server 23, passare dall'uso del B+Tree non cached a quello cached in indicizzazione differenziale ha ricondotto le prestazioni a quanto normalmente atteso, fare altrettanto con il BTree della serie 22 non ha sortito effetti tangibili. Il server spunta approssimativamente le stesse prestazioni precedenti senza sensibili miglioramenti. A parità di tempo si inserisce 1 documenti in più e se ne modificano 2.

Di fatto quello che si rileva, ora come ora, è che con archivi vuoti, un solo thread, intervallo tra un comando ed il successivo tra i 15 ed i 30ms e 180 secondi totali di attività, il server 23 spunta risultati molto simili al 22. Di fatto, usando il B+Tree si salva e si modifica un documento in meno mentre usando il B+Tree si salvano gli stessi documenti pur modificandone 2 in più. Visto così, su un test di 180 secondi, il B+Tree non appare una soluzione vincente rispetto al vecchio B+Tree.

Bisogna effettuare test più complessi e di durata maggiore per avere la sensazione della differenza comportamentale.

Riparliamo del Crash

Nonostante gli interventi e quindi nonostante l'uso del B+Tree cached anche in fase di indicizzazione differenziale, il Crash che si era presentato sin dall'inizio si ripresenta metodicamente. La manifestazione può differire. Se nel primo caso abbiamo visto una situazione in cui si cerca di introdurre in un nodo un elemento in più ritenendo che in esso ci sia ancora posto mentre risulta di fatto saturo e la dimensione immaginata è in realtà errata, esistono casi diversi in cui si proceder a fare split di un nodo che, valutandone il contenuto, risulta in realtà vuoto. Per contrprova sono stati effettuati test con molteplici thread (5 e 10) con un server 23 post natalizio ma partendo da un archivio che presenta ancora il file .idx e che quindi utilizza il BTree anziché il B+Tree. In tal caso non si verificano crash di alcun tipo. Per escludere che si tratti della cache, il test successivo, con l'uso del B+Tree, verrà effettuato senza l'ausilio della stessa.

No cache result

Escludendo la cache dall'uso del B+Tree il problema del crash non si presenta sia tentando con 5 che con solo 2 threads in parallelo. Va detto che, in assenza di cache, la lentezza del server è tale che non si può effettivamente affermare che non ci sia una fortuita combinazione di eventi che non causa il manifestarsi dell'errore (per lo meno questo è quanto non mi sento di affermare non conoscendo esattamente come si comporti il B+Tree). Di fatto, introducendo nuovamente l'uso della cache, l'errore si manifesta quasi istantaneamente. In breve, quindi, pur non avendo (ancora) la prova provata che il problema risieda nella gestione della cache, ho il forte sospetto che quello sia il punto in cui investigare più a fondo.

Un ulteriore test, arricchendo con molti logs lo stato delle cose, mi da un errore in un punto del tutto diverso, un punto legato alla ricerca.

Procedendo con un ulteriore test mi imbatto in una condizinoe che definirei curiosa. Sto cercando di inserire una chiave e non so dire se essa sia già presente o meno. La funzione bpt_insert_key() [bptfun.cpp] evoca la bpt→insert per la parola richiesta. La insert() giunge rapidamente a chiamare la findAdd() [BPTAlgorithms.h]. Nella findAdd() si fa un primo accesso ad un nodo. Esso non è foglia, se ne estrae il Link adatto e si evoca nuovamente la findAdd(). Al secondo giro di findAdd() anche questo nodo non risulta essere foglia, se ne estrare il link opportuno ma alla successiva chiamata alla writer.getNode() si scopre che il nodo non è valido.

```
         // Estrai link
        link = nodeRad->link(retIndex);
         NodeType *nodeL;
         // Verifico la correttezza
        if (!writer.getNode(&nodeL, link))return false;
```

La writer.getNode() torna 'false' e quindi causa un return 'false' anche alla findAdd() che riflette all'indietro questo comportamento causando alla ScaricaTreeIncr() o alla ScaricaTreeDiff() un'uscita con errore. Ne consegue che diverse operazioni di indicizzazione non vanno a buon fine perché risulta che si è verificato un errore di inserimento di una chiave nel B+Tree. Forse non ne avrò le prove provate, ma la convinzione che sia il trattamento della cache ad essere incriminato mi pare essere sempre più convincente.

Punto critico

La dimensione pari a 0 del nodo che viene splittato si è ripresentata e mi ha insospettito. Ad un analisi più accurata è risultato evidente che il puntatore al nodo che si stava splittando ed a quello splittato era lo stesso e che quindi, nel predisporre il nuovo nodo splittato, si era finito con il “piallare” quello esistente. Un esame più approfondito della cosa ha evidenziato come il membro 'addrNext' del FileController fosse… …fuori controllo!

Di fatto si procedeva ad evocare la newNode() che a sua volta evocava la getNextAddr(). L'indice di nodo risultante apparteneva già alla cache, cosa che non sarebbe dovuto accadere trattandosi di un nodo del tutto nuovo, e la successiva chiamata al metodo regenerate() 'piallava' il contenuto del nodo precedente, che guarda caso era proprio il nodo che si stava assumendo dalla cache via LRU. In altre parole, quel nodo che andava cercato nella cache per trovargli la nuova collocazione e che sarebbe dovuto risultare inesistente nella cache praticamente per definizione, risultava essere già presente in essa. Tutto questo è dovuto la mancato aggiornamento di 'addrNext' che viene caricato una tantum quando si apre il FileController poi viene ricalcolato di volta in

volta se il B+Tree viene usato senza l'ausilio della cache mente è il frutto di un semplice incremento in ogni altro caso. Chiaramente, quando più server convivono, nel momento in cui si passa dalla modalità 'non cached' a quella 'cached' si dovrebbe ricalcolare addrNext per gli usi successivi. Per sistemare quanto detto, quindi, è stato creato un nuovo metodo refreshAddrNext() che viene ora evocato:

1. In apertura del FileControlle, dove prima si procedeva a scoprire la size dello storage_.
2. Nel metodo newNode() dove, in assenza di cache, si valutava la nuova posizione sempre accedendo alla size dello storage_
3. Nel metodo openNodes() quando si passa in modalità cached così da avere un valore valido.

Il metodo è stato creato come privato in quanto non vi è ragione di renderlo disponibile ai fruitori del FileController essendo di suo esplicito utilizzo interno.

Test effettuati per 2 ore con 10 thread in parallelo hanno evidenziato la totale assenza di crash. Le condizioni d'errore rilevate sono le seguenti:

1. Un elevato numero di errori 808, ovvero errori di locking dovuti ad una naturale concorrenza d'accesso. Tali errori non destano alcuna preoccupazione (non dimentichiamoci che i test sono stati effettuati con un ritmo serratissimo, senza lasciar intercorrere che un intervallo brevissimo tra una unit e la successiva, testando più lo stress da carico che un comportamento umano credibile).
2. 6 errori 802, ovvero errori di accesso all'archivio. Essi sono intercorsi durante le ripartenze del test (di fatto si tratta di due o tre test da 10 minuti ed un successivo test da 2 ore) in cui un server appena partito non riesce a fare accesso all'archivio a causa di un locking pendente sullo stesso da parte di un altro server. In altre parole, anche questi errori sono naturale conseguenza di concorrenze d'accesso ed in quanto tali non destano preoccupazione.
3. 1 solo errore 810, riferito ad un file che non è stato possibile identificare. Esso si riferisce ad un file di selezione non più disponibile, presumibilmente rimosso a causa della sua vetustà.

Nuovi test di carico

Corretto l'errore appena descritto viene il momento di compiere nuovi test di carico che ci dicano la capacità effettiva del server di compiere inserimenti e modifiche con una batteria di server in parallelo. Riportiamo solo i valori più significativi e porremo in rapporto il server di classe 22, con la correzione inerente l'uso della cache anche in indicizzazione differenziale, ed il server di classe 23 post natalizio, quello al quale è stata riveduta la gestione dei files e delle loro scritture(riferite alle header) e flush. Ovviamente tutte le versioni utilizzate sono state compilate in modalità 'Release'.

Si ricorca che i valori di tempo riportati, con ogni probabilità, se pure indicati come massimi e minimi, si riferiscono al minimo e massimo tra i valori massimi rilevati dai diversi thread. Questo, per lo meno, sulla base delle impressioni raccolte con un thread solo. Ad ogni modo, in considerazione del fatto che la misurazioni vengono effettuate con lo stesso criterio, si confida che siano comunque significative.

I test vengono effettuati partendo ogni volta da archivi vuoti. La presenza di errori è naturalmente dovuta a condizioni di concorrenza d'accesso ed è a sua volta un utile indicatore.

Test 1.1

Condizioni di test: Tempo totale: 240”, Threads: 5, Intervallo: dai 15 ai 30ms, salvataggio XML.

Server 22	**Tot**	**Min**	**Max**	**Media**
Ricerche	**21**	113	397	255,66
Modifiche	**4**	1838	3730	2311,50
Inserimenti	**190**	2166	3897	2638,13
Mod post Ins	**187**	2126	3089	2475,34
Errori	**2**			
Server 23	**Tot**	**Min**	**Max**	**Media**
Ricerche	**20**	113	529	300,70
Modifiche	**4**	547	3170	1973,75
Inserimenti	**182**	585	3106	2511,31
Mod post Ins	**181**	1625	3180	2730,41
Errori	**3**			

Test 1.2

Condizioni di test: Tempo totale: 180“, Threads: 10, Intervallo: dai 50 ai 100ms, salvataggio XML.

Server 22	**Tot**	**Min**	**Max**	**Media**
Ricerche	**23**	278	1976	1164,65
Modifiche	**4**	3935	4798	4582,25
Inserimenti	**130**	2748	9931	4585,66
Mod post Ins	**125**	3678	15082	5745,95
Errori	**9**			

Server 23	Tot	Min	Max	Media
Ricerche	**25**	263	3266	1549,72
Modifiche	**4**	3104	4206	3440,75
Inserimenti	**138**	2287	10954	4633,13
Mod post Ins	**129**	1614	8802	5154,27
Errori	**11**			

Test 1.3

Condizioni di test: Tempo totale: 300”, Threads: 10, Intervallo: dai 2 a 3s, salvataggio XML.

Server 22	Tot	Min	Max	Media
Ricerche	**47**	540	3148	1490,65
Modifiche	**8**	3094	14300	7655,12
Inserimenti	**206**	2781	9772	4996,17
Mod post Ins	**192**	1880	13877	5729,74
Errori	**18**			

Server 23	Tot	Min	Max	Media
Ricerche	**32**	426	6157	2063,13
Modifiche	**9**	2951	20238	10799,22
Inserimenti	**157**	2766	15729	6566,37
Mod post Ins	**143**	3670	17544	7722,09
Errori	**23**			

Test 1.4

Condizioni di test: Le stesse del test numero 1.3 ma con un conteggio del tempo che si riferisce al solo comando e non anche al tempo necessario per scrivere il comando e leggere la risposta. Non vengono inoltre resettati i valori al variare dei threads.

Server 22	Tot	Min	Max	Media
Ricerche	**29**	307	2445	1279,58
Modifiche	**13**	1989	16521	4734,76
Inserimenti	**215**	3382	15245	4856,46
Mod post Ins	**204**	2331	7429	5260,16
Errori	**12**			

Server 23	Tot	Min	Max	Media
Ricerche	**62**	562	3632	1293,35
Modifiche	**8**	1492	5709	3545,87
Inserimenti	**208**	737	15089	4791,66
Mod post Ins	**197**	2621	15995	5733,34
Errori	**17**			

Un altro punto di vista

Visto che i risultati ottenuti sino ad ora non sono molto confortanti, anzi, non lo sono per nulla, vale la pena di compiere un altro test che mostri un punto di vista sostanzialmente differente. Ad essere messi a paragone sono lo stesso server, ovvero sempre la versione 23 post-natalizia che dovrebbe rappresentare la versione più avanzata del server disponibile e con tutti gli accorgimenti prestazionali più interessanti. La differenza nel test è che uno verrà effettuato su un archivio dotato di BTree e l'altro verrà effettuato su un archivio dotato di B+Tree.

I test vengono effettuati secondo alcuni criteri di cui al test numero 4 del precedente ciclo. Viene quindi preso in esame solo il tempo effettivo escludendo quello di IO.

Test 2.1

Condizioni di test: Tempo totale: 300“, Threads: 10, Intervallo: dai 2 a 3s, salvataggio XML.

Server 23BT	Tot	Min	Max	Media
Ricerche	**25**	212	2749	1238,24
Modifiche	**5**	2425	13631	6620,02
Inserimenti	**219**	2843	10028	4687,13
Mod post Ins	**211**	3040	12196	5303,18
Errori	**13**			

Server 23BPT	Tot	Min	Max	Media
Ricerche	**46**	580	3691	1511,00
Modifiche	**8**	2791	9264	6005,25
Inserimenti	**211**	2939	14957	4956,58

Server 23BPT	Tot	Min	Max	Media
Mod post Ins	**194**	2463	9461	5419,31
Errori	**20**			

Rientro dopo le ferie

La rientro di gennaio i test proseguono provvedendo ad introdurre nuve forme di benchmark ed a rimuovere ogni verifica riferita alle autorizzazioni per gli archivi dove questo non ha ragione di essere.

Test 3.1

Questo test viene effettuato per valutare quanto sia significativo il tempo impiegato dalla validazione delle attività da svolgere nel server 23 rispetto al server 22. Faremo nuovamente i testi di cui al punto 1.4 dopo aver modificato il server 23 perché ammetta una modalità *free access* al fine di verificare quanto sia l'impatto di tale validazione. Anche se potremmo basarci sui precedenti valori del test 1.4 sul server 22, li replichiamo comunque.

Condizioni di test: Tempo totale: 300", Threads: 10, Intervallo: dai 2 a 3s, salvataggio XML.

Server 22	Tot	Min	Max	Media
Ricerche	**38**	979	2025	1322,73
Modifiche	**13**	2140	13467	5271,30
Inserimenti	**220**	2292	15082	4467,81
Mod post Ins	**206**	2918	12793	5006,60
Errori	**15**			
Server 23	**Tot**	**Min**	**Max**	**Media**
Ricerche	**37**	176	2637	1495,00
Modifiche	**12**	2926	11019	5747,83
Inserimenti	**213**	3422	12032	4723,00
Mod post Ins	**204**	2024	14587	5481,91
Errori	**10**			

Si denota un miglioramento, se pur contenuto, del numero di inserimenti effettuati. Va però detto che c'è un minor numero di ricerche quindi il test potrebbe non essere altamente rappresentativo.

Cambiamo completamente direzione

I test effettuati sino ad ora sanno dirci che in condizioni randomiche i due server hanno comportamenti simili ma non uguali e che complessivamente *sembra* che il server 22 vada meglio del server 23. Per cercare di dissipare anche questo dubbio, procediamo con un test che si avvalga di WatchDoc. In questo modo le operazioni che i due server devono compiere sono del tutti identiche e si noterà, se pure solo in inserimento, il tempo di risposta dei due server.

Server	Documenti	Tempo
22	1657	265,797
23	1657	573,719
23idx	1657	258,203

Aiuto!!!! Il server 23 non *sembra* impiegare molto di più, lo fa! Inoltre lo fa perché produce un BTree che contiene una serie di nodi piuttosto corposa dai quali risultano un numero di chiavi molto superiore a quelli rilevati dal 22 o dal 23 con idx (49mila contro 37mila). L'esportazione delle chiavi mostra come esse siano in palese disordine. Il test, effettuato con il server 20090803 (ma che fa uso, ora, del B+Tree modificato negli ultimi giorni dell'anno per correggere il problema di switching) da lo stesso comportamento errato. Va ora compreso se il problema è in qualche modo riconducibile all'intervento compiuto sul B+Tree nei giorni tra natale e capodanno per risolvere il problema dello switching (ed il conseguente reload della posizione dell'ultimo nodo assegnato) oppure sia insito nell'attuale conformazione del codice.

Separazione dei problemi

Bisogna necessariamente distinguere e separare le questioni. Nello sviluppare interventi tesi a ridurre quanto più possibile il numero di letture e scritture non necessarie, e con esse le operazioni di flush, abbiamo rilevato altre problematiche prestazionali legate, a quanto pare, al B+Tree. Oltre agli aspetti prestazionali, inoltre, si è finito con il verificare problemi di affidabilità sempre del B+Tree. Questo ci impone di separare i due temi. Il progetto riferito agli interventi sulle scritture e sui locking va quindi chiuso e il resto delle attività dovrà afferire agli sviluppi relativi al solo B+Tree.

Test dopo l'identificazione dei diversi problemi

Atto primo

Effettuiamo quindi dei test conclusivi che ci dicano se gli interventi compiuti hanno sortito gli effetti desiderati o meno.
La condizione è quella già incontrata in precedenza: 10 Thread, tempo di intervallo tra i 2 ed i 3 secondi, tempo limite 300 secondi senza il conteggio del tempo necessario all'I/O dei comandi.

Server 23-pre	Tot	Min	Max	Media
Ricerche	**47**	638	6863	3165,48
Modifiche	**7**	6714	14952	11318,85
Inserimenti	**109**	1851	20765	9269,02
Mod post Ins	**83**	6066	22867	12771,36
Errori	**39**			

Server 23-post	Tot	Min	Max	Media
Ricerche	**52**	52	3227	1506,55
Modifiche	**17**	1801	11785	4827,52
Inserimenti	**207**	1207	8442	4490,30
Mod post Ins	**196**	2341	14826	5533,06
Errori	**20**			

Queste prime rilevazioni hanno lo scopo, in sostanza, di dimostrarci che i più recenti interventi effettuati sul server 23 pre-natalizio, al fine di produrre a sua volta logs di benchmark, non hanno inibito il funzionamento complessivo del server. Quello che ora andremo a confrontare sono proprio i risultati dei benckmark verificando, per altro, che i risultati delle operazioni svolte siano coerenti e che gli archivi siano "sani".
Malauguratamente quest'ultima affermazione non trova conferma. Nei test effettuati precedentemente non ci si era mai realmente preoccupati di verificare la salute dei B+Tree prodotti e le ricerche avevano dato risultati apparentemente coerenti. Tali risultati erano di fatto limitati alle chiavi che risultavano accessibili al vocabolario con procedure di posizionamento e letture *next*. Di fatto non vi era alcuna verifica sul fatto che il B+Tree risultante fosse congruente. Ci si accontentava, in due parole, del fatto che non si producessero condizioni di crash.
Ora i test vanno replicati con un *thread* solo per verificare se sia la concorrenza d'accesso o l'uso base del B+Tree ad essere fonte del problema. Questo, comunque, fa parte del 2° problema. Il primo, relativo agli accessi su disco ed alla validità delle header lette e scritte, si completa con la verifica dei log di benchmark prodotti.

I valori evidenziati

Veniamo quindi ai valori evidenziati dal precedente test.

```
Il numero medio di aperture, posizionamenti, letture e così via si riferisce solo alle operazioni
di inserimento e modifica.
```

Vengono posti in evidenza i risultati più significativi.

Descrizione/Server	23-pre	23-post
Operazioni Totali	1261	2159
Tempo Medio Ricerca	47,23	88,83
Tempo Medio Generale	1115,55	664,07
Salvataggi Totali	199	419
Tempo Medio Salvataggi	6869,90	3094,45
Numero Medio Aperture	2,90	3,41
Numero Medio Posizionamenti	1016,30	328,54
Numero Medio Letture	423,80	142,30
Numero Medio Scritture	319,18	170,23
Numero Medio *Flush*	6,44	6,79
Numero Medio *Lock/Unlock*	4,75	12,65

Atto secondo

Per dar maggior consistenza ai test appena effettuati essi vengono replicati esattamente con gli stessi server ma con archivi basati su BTree anziché sul B+Tree, non tanto per valutare la differenza comportamentale ma per stabilire che essi portano ad archivi corretti. Ci si aspetta quindi che gli accessi al vocabolario per le ricerche così come per i salvataggi dei documenti risultati ancora più affidabili.
La condizione di test è, ovviamente, la stessa di quello precedente.

Server 23-pre	Tot	Min	Max	Media
Ricerche	**36**	504	2235	1403,75
Modifiche	**10**	2902	6374	4726,70
Inserimenti	**212**	1839	8508	4838,20
Mod post Ins	**205**	1759	8567	5170,42
Errori	**8**			

Il controllo vocabolario al termine delle operazioni ha dato esito corretto.

Server 23-post	Tot	Min	Max	Media
Ricerche	**44**	494	3516	1412,06

Server 23-post	Tot	Min	Max	Media
Modifiche	**5**	1904	17348	5445,20
Inserimenti	**229**	2310	11398	4026,73
Mod post Ins	**223**	4243	8674	5310,67
Errori	**9**			

Il controllo vocabolario al termine delle operazioni ha dato esito corretto.

I valori evidenziati

Veniamo quindi ai valori evidenziati dal precedente test.

```
Il numero medio di aperture, posizionamenti, letture e così via si riferisce solo alle operazioni
di inserimento e modifica.
```

Vengono posti in evidenza i risultati più significativi ed in minor evidenza quelli degni di nota se pure non eclatanti come gli altri.

Descrizione/Server	23-pre	23-post
Operazioni Totali	1971	2163
Tempo Medio Ricerca	53,42	72,43
Tempo Medio Generale	701,73	651,45
Salvataggi Totali	427	457
Tempo Medio Salvataggi	3131,60	2920,43
Numero Medio Aperture	3,45	3,45
Numero Medio Posizionamenti	1263,70	350,06
Numero Medio Letture	530,89	159,99
Numero Medio Scritture	406,99	179,94
Numero Medio *Flush*	8,83	7,84
Numero Medio *Lock/Unlock*	5,89	12,75

Diverso test

Replichiamo anche i casi in cui il test ha luogo su un archivio vuoto (di tipo agi) sul quel viene inserito un numero preciso e sempre uguale di documenti.
In questo modo i test effettuati su due diversi server dovrebbero essere maggiormente rappresentativi.

Nel momento in cui questi test vengono effettuati il server 23 con B+Tree produce ancora comportamenti imprecisi, per questa ragione il test viene effettuato solo sul BTree pur sapendo che il B+Tree, nei test poco affidabili realizzati sino ad ora, spunta dei tempi molto peggiori.

Server	Documenti	Tempo	Aperture	Posizionamenti	Letture	Scritture	Flush	Lock
23-pre-bt	1657	258,187	4.982	2.073.439	932.557	569.289	4.978	6.629
23-post-bt	1657	290,437	4.982	443.140	97.988	321.964	3.321	13.269

Il server post-natalizio compie una quantità di I/O assolutamente differente ed in questo modo dovrebbe ottenere risultati estremamente migliori ed invece impiega un tempo superiore.
A questo punto vale la pena di valutare quanto tempo spendiamo per i locking che ora facciamo ampiamente sulle headers.

Dopo aver compiuto interventi tesi proprio ad identificare i casi di lettura e scriuttura ma ancor più di locking più rappresentativi nelle operazioni di acquisizione documenti, riconducendo il comportamento del server al suo comportamento originario, i test vengono effettuati nuovamente nelle stesse condizioni.

Poiché ad una prima verifica ci siamo resi conto che non si erano ottenute variazioni significative, sono stati effettuati ulteriori interventi di perfezionamento.

Server	Documenti	Tempo	Aperture	Posizionamenti	Letture	Scritture	Flush	Lock
23-pre-bt	1657	291,078	4.982	2.073.439	932.557	569.289	4.978	6.629
23-post-bt 1° intervento	1657	268,968	4.982	443.140	97.988	321.964	3.321	13.269
23-post-bt 2° intervento	1657	265,578	4.982	443.140	97.988	321.964	3.321	8.298
23-post-bt 3° intervento	1657	263.031	4.982	443.140	97.988	321.964	3.321	6.629

La sorpresa che ci riservano questi test è che, nonostante una quantità di letture e scritture immensamente più piccola e riconducendo il numero dei flush a quello pre-natalizio, gli interventi che abbiamo compiuto non mostrano di avere portato alcun beneficio o, per lo meno, benefici non tangibili. Non dimentichiamo che nelle stesse condizioni il test effettuato pochi giorni prima aveva dato rilevamenti temporali inversi o quanto meno aveva mostrato che il server pre-natalizio poteva ottenere gli stessi risultati.
In pratica non v'è alcun beneficio.
Il test successivo procede ad una riduzione dei flush, eliminando i flush dovuti ai titoli.

Server	Documenti	Tempo	Aperture	Posizionamenti	Letture	Scritture	Flush	Lock
23-post-bt flush title weak	1657	78,781	4.982	443.140	97.988	321.964	7	6.629
23-post-bpt flush title weak	1657	485,015	4.982	443.140	97.988	321.964	6	6.629

Questo ci mostra che il B+Tree, **ora funzionante correttamente**, compie flush ad ogni chiusura della cache e quindi il suo comportamento non è paragonabile.
Torniamo quindi a test effettuati con SoapUI che ci mostrino una media comportamentale credibile.

Considerazioni

Il confronto tra i due test effettuati in chiusura di questo sviluppo evidenziano come la riduzione in quanto ad operazioni di lettura, scrittura e posizionamento sia decisamente tangibile. Essa si fa sentire maggiormente nel caso di B+Tree e meno nel caso del BTree ma pone comunque in evidenza che le operazioni di salvataggio avvengono mediamente in un tempo inferiore compiendo molte meno operazioni di accesso al disco.
Cresce, di fatto, il numero di *Lock/Unlock* in quanto ne sono stati aggiunti per ragioni di maggior sicurezza ma non si può imputare ad essi un effettivo degrado prestazionale. In effetti le copie precedenti del server non compivano il lock in lettura per i server che non erano tenuti a rinnovare effettivamente le headers. Ci si limitava quindi a compiere una semplice lettura senza locking. Ora che la gestione dei locking è stata compiuta in modalità *smart* dalla funzione stess di locking, si compiono effettivamente più operazioni di locking.

Il server 23 pre e post natalizio ha un comportamento corretto se utilizzato con il BTree mentre nello sviluppo del B+Tree dev'essere presente ancora qualche problema significativo ma, ragionevolmente, semplice da identificare.
Vista l'irregolarità dei risultati è ancora presto per dire se si sia ottenuto un qualche vantaggio dalla versione 22.