Corso di Laurea triennale

in INGEGNERIA INFORMATICA

Tesina di Laurea

Il Backup incrementale di una Working Directory realizzato in Python

di Guido Panzetta

Anno Accademico 2020/2021

Indice

Backup e Backup incrementale di una Working Directory

L’Applicazione:

- Linguaggio utilizzato

- Analisi dell’algoritmo

- Utilità dell’applicazione

Repository e GitHub

**BACKUP INCREMENTALE DI UNA WORKING DIRECTORY**

Un progetto software, sviluppato da un gruppo di lavoro, è solitamente composto da molti moduli, librerie e centinaia di programmi. Necessita di continui aggiornamenti che devono essere poi integrati. A volte questo processo risulta assai complesso. Diventa indispensabile effettuare regolarmente il backup ovvero la duplicazione del contenuto di un sistema informatico memorizzato su un supporto esterno (disco rigido o chiavetta USB), in un server **NAS** (Network Attached Storage), cloud oppure online.

Tra le varie tipologie, il backup incrementale utilizza una precedente copia del sistema, completa o essa stessa incrementale e va a duplicare e salvare solo i file nuovi e quelli modificati. Lo spazio richiesto è minore, il salvataggio dei dati diviene facile da gestire e l’aggiornamento risulta veloce.

In tal senso ho realizzato un’applicazione che effettua il backup incrementale di una Working Directory.

Il linguaggio di programmazione utilizzato per l’algoritmo è il Python, che mette a disposizione, tra i vari tipi di dato, i *dizionari*, sequenze di elementi caratterizzati da coppie chiave-valore.

L’idea alla base del progetto è stata generare, al lancio dell’applicazione, un dizionario (*index*) che contenesse come chiavi le absolute path dei file che fanno parte della working directory e come valori associati i relativi rifermenti temporali di creazione e modifica. L’utilizzo del dizionario in memoria rende l’algoritmo molto veloce e la sua utilità verrà mostrata nel seguito.

Vediamo meglio quale sia il funzionamento e la logica dell’applicazione.

**PRIMA ESECUZIONE: CREAZIONE DELL’AMBIENTE DI LAVORO E DEL PRIMO DIZIONARIO**

La prima volta che si esegue il codice e si esplora la Working Directory sono creati:

* una ***Backup\_Directory* locale**, ove è riprodotta la struttura ad albero della *Working\_Directory* ed ivi copiati tutti i file in essa contenuti, nelle rispettive sottocartelle. Inizialmente viene prodotto un backup totale del sistema. La prima esecuzione pertanto è molto più lenta delle successive.
* un **dizionario** (*index*) che ha per chiavi le absolute path dei file che compongono la Working Directory e per valori i riferimenti temporali di creazione e modifica. I dati ivi contenuti vengono scritti in un file *(index.gz),* compresso e serializzato all’interno della Backup Directory in locale.

**SUCCESSIVE ESECUZIONI DELL’APPLICAZIONE**

Ad ogni lancio successivo dell’applicazione, al momento di effettuare un nuovo backup incrementale, il file index.gz, che si trova nella *Backup\_Directory*, viene aperto in lettura e scompattato[[1]](#footnote-1). L’algoritmo genera e carica in memoria, con i dati letti dal file, un dizionario composto da elementi chiave-valore (le absolute path dei file della Working\_Directory ed i relativi riferimenti temporali di creazione e modifica che i file avevano alla data del precedente backup[[2]](#footnote-2).

**ESPLORAZIONE DELLA WORKING DIRECTORY**

Durante l’esecuzione dell’algoritmo viene esplorata l’intera Working Directory attuale[[3]](#footnote-3), che probabilmente, nelle varie sottodirectory, contiene nuovi file oltre a quelli già esistenti. Taluni di questi ultimi[[4]](#footnote-4) possono essere stati modificati ed avere quindi dei riferimenti temporali di modifica differenti e successivi rispetto ai medesimi file memorizzati nel backup precedente[[5]](#footnote-5) e di conseguenza anche rispetto ai valori delle chiavi del dizionario index in memoria, costruito con i dati del file index.gz serializzato alla data del precedente backup.

L’algoritmo, pertanto, verifica che ogni file, incontrato nell’esplorazione della Working Directory, sia presente come chiave (la absolute path del file) del dizionario in memoria:

* **in caso negativo** il file incontrato è **NUOVO** e viene inserito, come elemento chiave-valore, nel dizionario attualmente in memoria e copiato sia nella Backup Directory sia nella sottocartella della Incremental Backup Directory, in posizione corretta.
* **in caso positivo** il file incontrato **NON È NUOVO** e nella cartella di Backup ne esiste uno con la stessa absolute path. Per verificare se il file incontrato nell’esplorazione della Working Directory sia stato modificato dall’ultimo backup, si confronta la data della sua creazione e ultima modifica con il valore corrispondente alla chiave (nome assoluto del file) presente nel dizionario in memoria[[6]](#footnote-6). In caso di modifica dall’ultimo backup il riferimento temporale presente nel dizionario (il valore della chiave) sarà antecedente e dovrà essere aggiornato con quello attuale. Copia dell’ultima versione del file deve essere memorizzata nella di Backup[[7]](#footnote-7) in posizione corretta.

**TERMINE DELL’ESECUZIONE DELL’APPLICAZIONE**

Prima del termine dell’esecuzione il dizionario, con le chiavi e i valori associati aggiornati alla data attuale, viene scritto nel nuovo file index.gz[[8]](#footnote-8), che viene ancora una volta compattato e serializzato[[9]](#footnote-9) e reso disponibile successivamente per un nuovo ciclo di backup.

La Backup Directory è stata aggiornata e contiene i nuovi file e quelli modificati alla data del backup attuale.

**UTILITA’ DELL’APPLICAZIONE**

L’utilità dell’applicazione può essere individuata nella velocità di individuazione dei file nuovi o modificati, tramite il dizionario in memoria creato con i dati del file index.gz. Questa modalità di selezione è molto più rapida rispetto ad un controllo tramite accesso su disco, seppur in locale. Questa è la vera utilità dell’applicazione.

**EFFICACIA DELL’APPLICAZIONE: SIMULAZIONI**

Le simulazioni dell’esecuzione dell’algoritmo di backup incrementale, effettuate su una working directory contenente molte subdirectory e file, mostrano la rapidità nell’individuare i file nuovi e modificati. Ad eccezione della prima[[10]](#footnote-10), le successive esecuzioni risultano molto rapide.

Un semplice algoritmo ne rileva la durata di esecuzione.

start = time.time()

end = time.time()

durata = end – start

Da queste verifiche si può concludere che …………………….

**REPOSITORY SU GITHUB**

Uno strumento ormai indispensabile per gestire il c.d. *versioning* è il repository[[11]](#footnote-11).

GitHub, come altri analoghi strumenti, offre uno spazio gratuito fondamentale per gestire processi di sviluppo del software. Ad ogni “commit” e “push” in aggiornamento dei programmi o dei singoli moduli segue la “pull” degli altri collaboratori, che possono operare anche da remoto e integrare i vari moduli nel progetto.

1. index = pickle.load(gzip.open(indexFile, "rb")) [↑](#footnote-ref-1)
2. Se fosse la seconda esecuzione dell’algoritmo il dizionario conterrebbe le absolute path e i riferimenti temporali di creazione e modifica dei file che sono stati memorizzati nel backup totale effettuato la prima volta. Se fosse una successiva esecuzione il dizionario conterrebbe le absolute path e i riferimenti temporali dei file che compongono la Working Directory alla data del precedente backup incrementale. [↑](#footnote-ref-2)
3. Alla data attuale di esecuzione dell’applicazione. [↑](#footnote-ref-3)
4. Nel corso dello sviluppo del sistema software che si sta creando o per il semplice utilizzo della working directory come contenitore di file prodotti in altri modi e di cui si ha necessità di “backuppare”. [↑](#footnote-ref-4)
5. Serializzato e compattato nella Backup Directory. [↑](#footnote-ref-5)
6. Che avrà i riferimenti temporali di ultima modifica dello stesso file alla data del precedente backup. [↑](#footnote-ref-6)
7. Nella Backup Directory l’ultima versione del file deve sostituire quella precedente. [↑](#footnote-ref-7)
8. Che va a sostituire il vecchio file [↑](#footnote-ref-8)
9. pickle.dump(index, gzip.open(indexFile, "wb")) [↑](#footnote-ref-9)
10. La prima esecuzione crea il backup totale della working directory e può risultare dispendiosa in termini di tempo a causa di tutte le operazioni di copy necessarie [↑](#footnote-ref-10)
11. Da Wikipedia un repository (letteralmente deposito o ripostiglio), in informatica, è un ambiente di un sistema informativo (ad esempio di tipo ERP), in cui vengono gestiti i metadati, attraverso tabelle relazionali; l'insieme di tabelle, regole e motori di calcolo tramite cui si gestiscono i metadati prende il nome di metabase. [↑](#footnote-ref-11)