**Compressione e decompressione timeline!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!**

**Divisione compiti in timeline!!!!!!!!**

**Ottimizzazazione codice**

**Snappy, l’algoritmo del compromesso / La rivincita del più lento/dell’LZ77**

Immaginate di avere centinaia o migliaia di computer che devono poter trasferire ingenti moli di dati dall’uno all’altro il più velocemente possibile. Capite che la velocità di trasmissione è proporzionale alla dimensione dei dati. Viene spontaneo quindi pensare dii comprimere le informazioni prima di inviarle. Tuttavia se il tipo di compressione ha un costo, di risorse e di tempo, troppo elevato ti puoi ritrovare nella situazione in cui hai peggiorato le cose invece di migliorale.

Si tratta chiaramente di un problema di ottimizzazione in cui bisogna trovare uno sweet spot, il giusto tasso di

E chi se non Google poteva trovare una soluzione efficace a questo problema. Così è nato Snappy, algoritmo reso open source nel 2011. Come suggerisce il nome, Snappy punta tutto sulla velocità. Google afferma che Snappy è in grado di raggiungere tranquillamente una velocità di 250 MB/s in compressione e 500 MB/s in decompressione, mentre i file compressi sono tra il 20 e il 100% più grandi delle altre librerie di compressione disponibili. Ideale quindi per tutte quelle situazioni in cui a fare da collo di bottiglia sono le risorse di CPU e non quelle di storage (che poi sono la maggior parte dei casi).

Ma come funziona Snappy? Ad essere sincero ci è risultato davvero difficile trovare della documentazione valida, colpa soprattutto di quanto sia recente l’agoritmo. Alla fine abbiamo trovato quello che cercavamo in una combinazione di tesi di master di altre università che ci davano una quadro sul funzionamento di Snappy*,* e del codice e documentazione ufficiale che ci ha permesso di confermare le nostre ipotesi.

Cosa abbiamo scoperto?

L’algoritmo funziona così: la dimensione del file non compresso viene scritta in formato varint, di cui vi parlerò tra poco, all’inzio del file compresso. Questo permette al decompressore di conoscere quanto grande sarà il file una volta decompresso. Successivamente il file da comprimere viene diviso in blocchi di stessa dimensione, 32kB, che vengono compressi in maniera indipendente l’uno dall’altro. Questo vuol dire che ogni blocco non ha memoria di quello precedente.

Snappy ha alla base l’algoritmo di compressione LZ77 e speroche questo sorprenda voi quanto ha sorpreso noi, visto che nella sua versione base è conosciuto per la sua lentezza di compressione (tra quelli visti in classe è il più lento).

Come fa quindi Snappy a essere così veloce?

Beh, vi ricordate che alla base dell’Lz77, c’è la ricerca della sequenza corrente nella finestra di dati già processati. Ed è proprio questo processo di ricerca che può risultare molto lento. Snappy invece ne fa il suo punto di forza.

Infatti il processo di ricerca di match viene effettuato solo per gruppi di 4 bytes e utilizzando un’hash table. In pratica, attraverso una funzione hash, ad ogni gruppo di 4 bytes viene associato un hash code che costituisce l’indice di accesso all’hash table. Ad ogni posizione dell’array si trova un offset. È possibile scoprire o che si tratta di una sequenza mai vista prima e quindi salvarne la posizione, o che si tratta di una copia di cui possiamo ritrovare la posizione grazie all’offset. Una volta che si è trovato un riscontro, si controlla prima se la corrispondenza può essere più lunga di soli 4 bytes e successivamente le informazioni sulla copia (lunghezza e offset) vengono messe in output. E da lì si ricomincia. Vedete che la ricerca di un match avviene a tempo costante.

È un formatto che permette di esprimere in maniera efficiente dei numeri di cui in partenza non si conosce l’ordine di grandezza. Infatti sacrifica il MSB di ogni byte per utilizzarlo come segnale

Sweet spot

Grafico compression/speed IO

E chi se non Google poteva avere bisogno di una soluzione efficace

Ricerca: Partendo da…..

LZ77 così lento? Perché non un altro algoritmo?

* Hash table

FORMAT:

Varint 🡪 perché lunghezza proporzionle al numero

Copie 🡪 Perché 3 tipi di copie?

Quindi decompressione.

Organizzazione del lavoro

Per concludere piccolo riassunto che espone….

3w dicembre 🡪 Compressione e decompressione capaci di produrre e leggere snappy format

15 dicembre 🡪 test, solve bugs, ottimizzazione codice. Ci aspettiamo molti problemi

22 dicembre 🡪 codice documentato, relazione tecnica