Sistemi Operativi – Prova del 23 Settembre 2011 - Tempo a disposizione 3.5 ore

Si deve scrivere del codice parallelo specializzato per la compressione di immagini residenti su file. Si può assumere che ogni immagine non compressa sia codificata come sequenza di linee (terminate dal carattere di fine linea '\n'), ognuna delle quali contiene una sequenza di interi separati da spazi. Ad esempio,

```
10 10 10 4 4 5 5 5 5 5
3 3 3 3 1 1 1 1 1 12 12 4
…
73 73 73 3 3 1 3 3 3 3
```

Rappresenta un possibile file immagine non compresso. I file immagine devono essere compressi eliminando le ripetizioni: una sequenza N occorrenze del numero X deve essere rimpiazzata con il valore 'N X'. Ad esempio il file immagine di esempio diventerebbe:

```
3 10 2 4 6 5
4 3 5 1 2 12 1 4
…
3 73 2 3 1 1 5 3
```

Si progetti la classe **Compressore,** dotata del metodo pubblico

```
comprimi( string pathfilesorgente, string pathfiledestinazione)
```

tale metodo deve aprire `pathfilesorgente`, che si assume nel formato 'immagine non compressa' specificato sopra, e scriverne il contenuto nel formato compresso specificato su `pathfiledestinazione.`
La fase di compressione **DEVE essere implementata in multi-threading ottimizzando le prestazioni temporali e assumendo di avere a disposizione un processore con 8 CPU.**

Per la gestione dei file si usi la classe `fstream` e i corrispondenti metodi e operatori `open, close, ">> " e "<<"` (dal significato intuitivo).

*E' **parte integrante** della prova di esame completare le specifiche date nei punti non esplicitamente definiti, introducendo tutte le strutture dati che si ritengano necessarie, e risolvendo eventuali ambiguità. Non è consentito modificare il prototipo dei metodi se questo è stato fornito, mentre è possibile estendere le classi con tutti i campi e metodi privati che si ritengano necessari.*
*La scelta della strategia di parallelismo e di coordinamento tra i thread più opportuna è **A CARICO DELLO STUDENTE.***

Alcuni suggerimenti su come si poteva svolgere la traccia
Un modo per impostare questa traccia può essere quello di allocare un certo numero di Thread Worker paralleli, i quali prelevano le righe da comprimere, per poi depositarle in un buffer, che chiameremo OUT, dove vengono depositate le righe compresse. Prima di scrivere le righe compresse sullo stream di uscita, queste devono essere riordinate nella stessa sequenza con cui sono state lette da input: non è infatti garantito che i thread Worker depositino le righe compresse nell'ordine corretto su OUT. Il problema può

essere risolto associando un numero progressivo univoco alle linee da comprimere, e progettando opportunamente un thread Riordinatore che legga da OUT nella maniera opportuna. Si noti che OUT non potrà essere un normale buffer FIFO, ma dovrà consentire la possibilità di accedere casualmente ai singoli elementi presenti nel buffer.

Gli errori più comuni commessi da chi si è cimentato con questa traccia

1. **Leggere tutto il file in memoria**, allocare dei thread worker suddividendo il carico sui dati caricati, **tenere memorizzate le righe compresse in memoria,** e infine attendere che i worker si sincronizzino su una barriera. Dato il particolare tipo di problema, questo approccio suicida porta a un enorme spreco di tempo (il file viene caricato, compresso e scritto in tre tempi diversi!) e memoria (bisogna tenere in RAM una copia del file di input e una copia del file in via di elaborazione!). La barriera non era il martello adatto per questo chiodo.
2. **Utilizzare un buffer di uscita OUT con normale politica FIFO.** Così facendo le righe compresse vengono scritte sul file di uscita *in ordine del tutto casuale.* E' necessario modellare come si deve i metodi messi a disposizione da OUT e il thread che lo svuota!