INDEXING: DISTRIBUITO O MAPREDUCE?

Davide Tuccilli

Algoritmi Avanzati

A.A. 2011/2012

Introduzione

- Grandi moli di dati su cui effettuiamo ricerche quotidianamente
- Google indicizza per il motore di ricerca circa 20TB di dati
- Mole di dati tale da dover essere distribuita

Comparing Distributed Indexing: To MapReduce or Not?

Richard M. C. McCreadie, Craig Macdonald, Iadh Ounis

- L'indexing nel distribuito scala linearmente se i dati si trovano sui nodi
- Collo di bottiglia nelle comunicazioni di rete
- MapReduce utilizza file system distribuiti (GFS, HDFS)
- Scheduling sfrutta la località dei file

Obiettivi dell'indexing

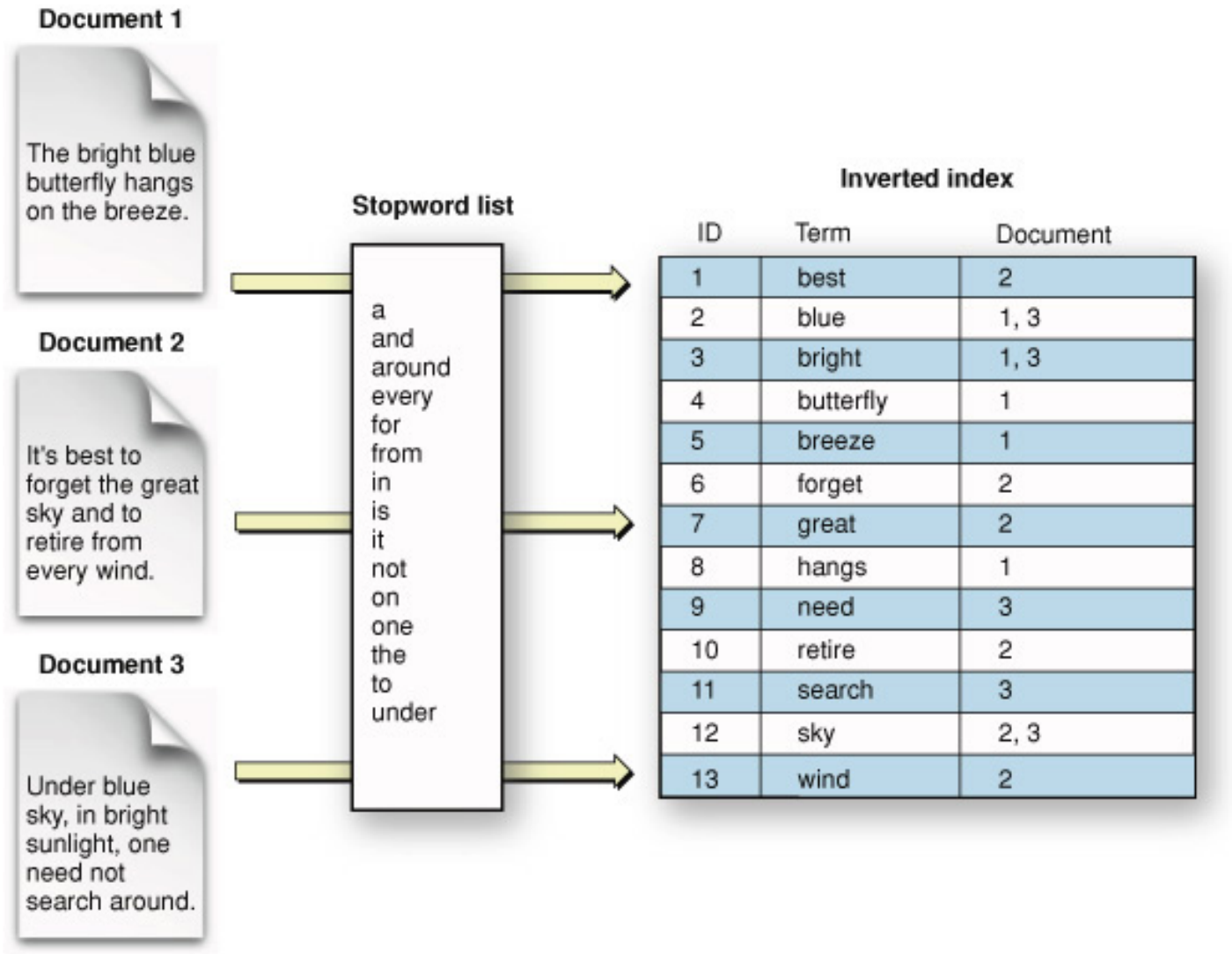
- Rendere veloce la ricerca di parole su un corpus
- Costruire una struttura dati che contenga le informazioni necessarie alla ricerca
- Rendere efficiente sia la costruzione della struttura dati sia la ricerca su di essa

Inverted index

- Associa i termini di un vocabolario alla lista di documenti del corpus in cui tale parola è contenuta

- Posting list corredate di informazioni per effettuare ranking e ricerca
  - Frequenza
  - Posizione delle occorrenze
  - Campo in cui si trova (titolo, grassetto, …)

Inverted index



ID	Term	Document
1	best	2
2	blue	1, 3
3	bright	1, 3
4	butterfly	1
5	breeze	1
6	forget	2
7	great	2
8	hangs	1
9	need	3
10	retire	2
11	search	3
12	sky	2, 3
13	wind	2

L'indexing di Dean e Ghemawat

"The map function parses each document, and emits a sequence of $(word,\ document\ ID)$ pairs. The reduce function accepts all pairs for a given word, sorts the corresponding document IDs and emits a $(word,\ list(document\ ID))$ pair. The set of all output pairs forms a simple inverted index. It is easy to augment this computation to keep track of word positions."

MapReduce: Simplified Data Processing on Large Clusters

L'indexing di Dean e Ghemawat

```
function MAP(String key, String value)
    docID = key
    docContent = value
    for all term ∈ docContent do
        term = Stem(term)
        if term isn't a stopword then
            Emit(term, docID)
        end if
    end for
    return x
end function
```

Un indexing più efficiente

- La strategia di Dean e Ghemawat prevede che i mapper diano in output ogni coppia $(term,\ document\ ID)$
- Ridurre il numero di coppie in output dai mapper potrebbe migliorare le prestazioni
- Caricare di un numero maggiore di informazioni ogni output

Apache Nutch

- Parte della piattaforma di IR di Apache
- Contiene un'implementazione di indexing basata su Hadoop
- Mapper danno in output coppie $(docID, analysedDocument)$
- Output lista di termini e frequenze di un documento
- I reducer devono invertire

Un approccio diverso

- Le coppie $(docID, analysedDocument)$ sono considerevolmente meno di quelle $(term,\ document\ ID)$
- Passo in avanti a livello prestazionale
- Il compito dei reducer diventa più complesso
- Obiettivo: sfruttare meglio il distribuito dando in input ai reducer delle posting list parziali

Single-Pass Indexing

- Nel modello External Memory, scrittura bufferizzata delle posting list
- In ogni momento in memoria troviamo le posting list parziali
- Merge delle posting list parziali in una inverted index
- Compressione online con Elias' gamma code

Elias' gamma code

- Codice prefisso che rappresenta un numero x con $2\lfloor \log_2 x \rfloor + 1$ bit

- Semplice da codificare e decodificare
  - Scrivi il numero che vuoi rappresentare in binario
    - Esempio: $14 = 1110$
  - Sia N il più grande esponente per cui $2^N \leq x$, si aggiunga un prefisso di N zero
    - Esempio: $2^3 \leq 14 < 2^4 \Rightarrow N = 3$, dunque $14 = 0001110$
  - Per decodificare contiamo i primi N zero, e decodifichiamo il numero binario sapendo che la prima cifra significativa corrisponde a 2^N

- Compressione efficace per rappresentare numeri piccoli

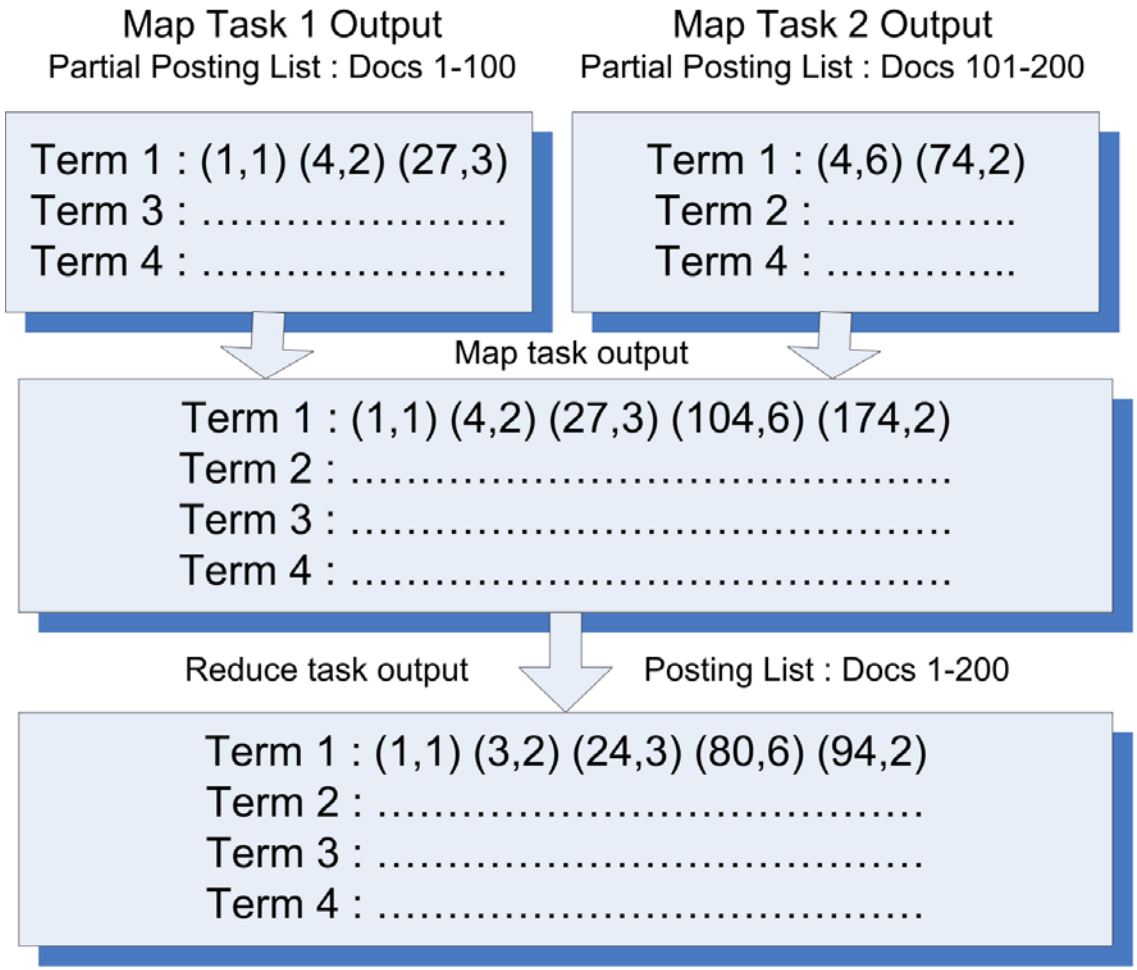
Single-Pass Indexing distribuito

- Inverted index divisa in frammenti distribuiti
- Si partizionano e distribuiscono i documenti da indicizzare a k inverter
- Gli inverter utilizzano l'algoritmo di inverted index per creare la propria inverted index
- Query a loro volta distribuite

Single-Pass Indexing in MapReduce

- Mapper ricevono una lista di documenti
- Output posting list parziali, effettuando un'operazione Emit ogni volta che esauriscono la memoria
- I reducer possono creare le inverted index
- Permette di mantenere lo stesso schema di compressione

Single-Pass Indexing in MapReduce



Test

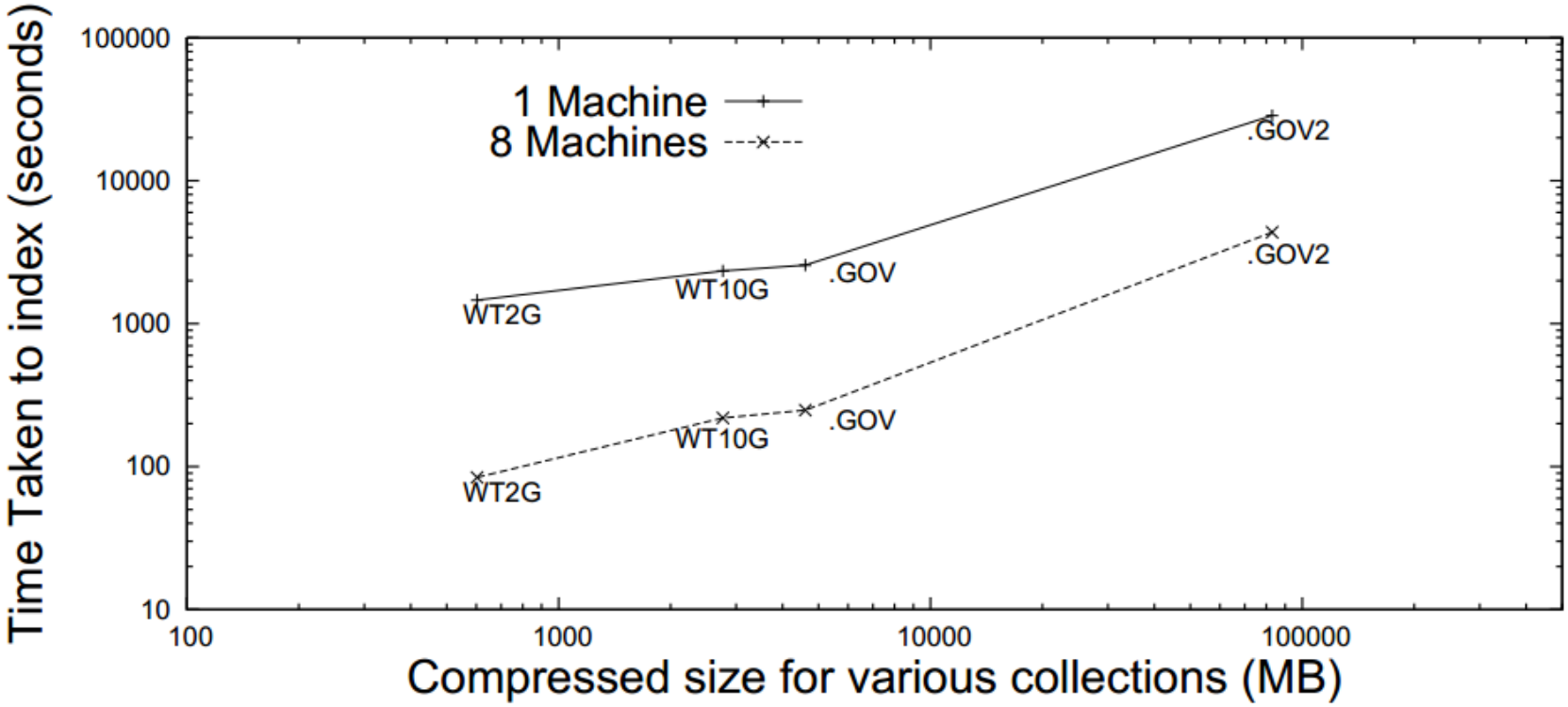
- Effettuati su piattaforma Hadoop con HDFS
- Corpus TREC GOV2
  - Crawl di siti .gov del 2004
  - Contiene 25,205,179 documenti
  - 0.42TB di dati non compressi
- Cluster di 19 macchine identiche
  - Intel Xeon 2.4GHz (a 4 core, di cui 1 lasciato libero per il sistema)
  - 4GB di RAM
  - Un HDD da 160GB @ 7200rpm con buffer a 8MB
  - Due HDD da 400GB @ 7200rpm con buffer a 16MB

Throughput

# Macchine (Cores)	1 (3)	2 (6)	4 (12)	6 (18)	8 (24)
Distribuito (Terrier IR)	2.44	4.6	12.8	12.4	12.8
Dean & Ghemawat	1.15	1.59	4.01	4.71	6.38
MapReduce Single-Pass	2.59	5.19	9.45	13.16	17.31

$$Throughput = \frac{Corpus\ size\ in\ MB}{Execution\ Time\ in\ secondi}$$

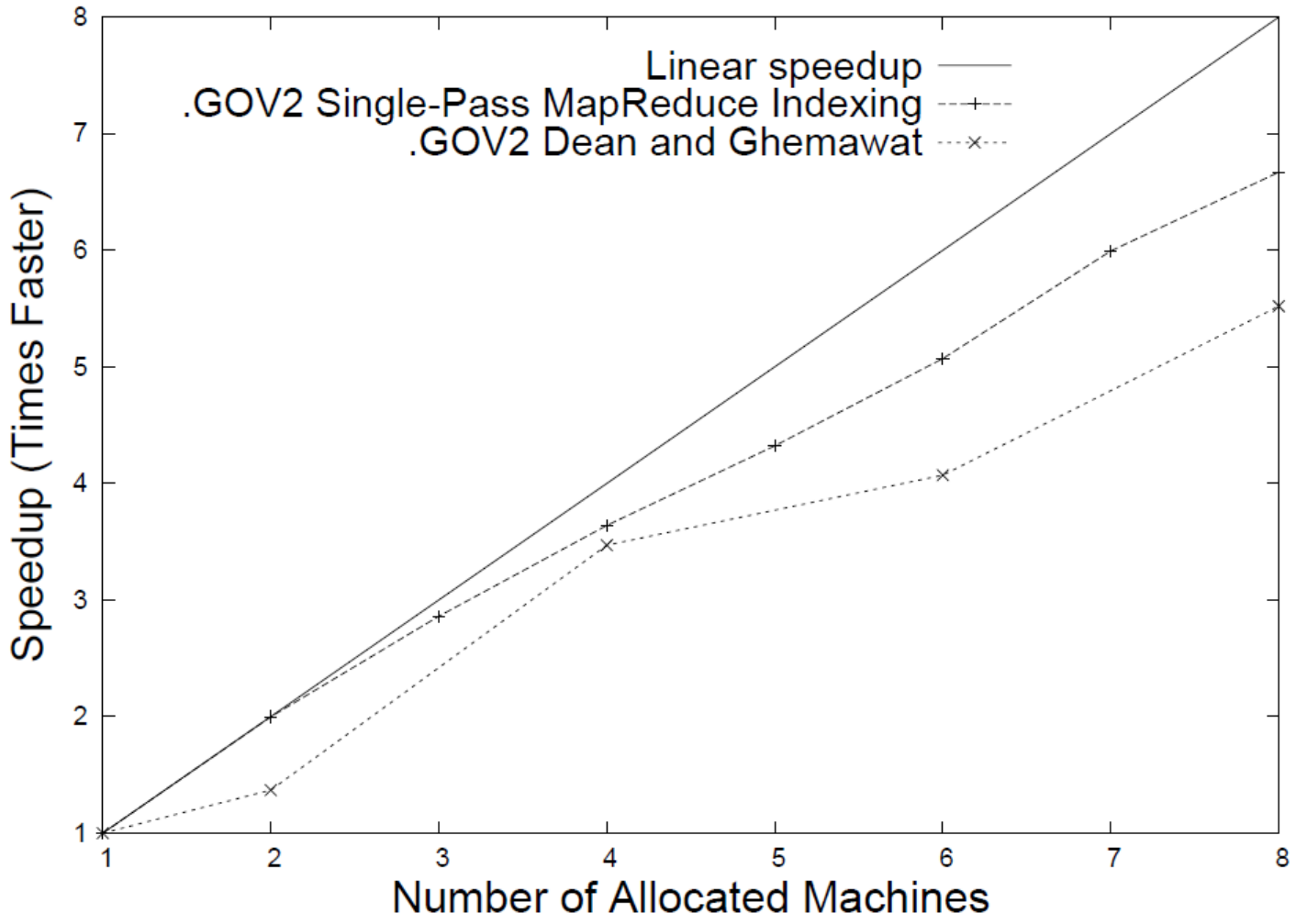
Scalabilità



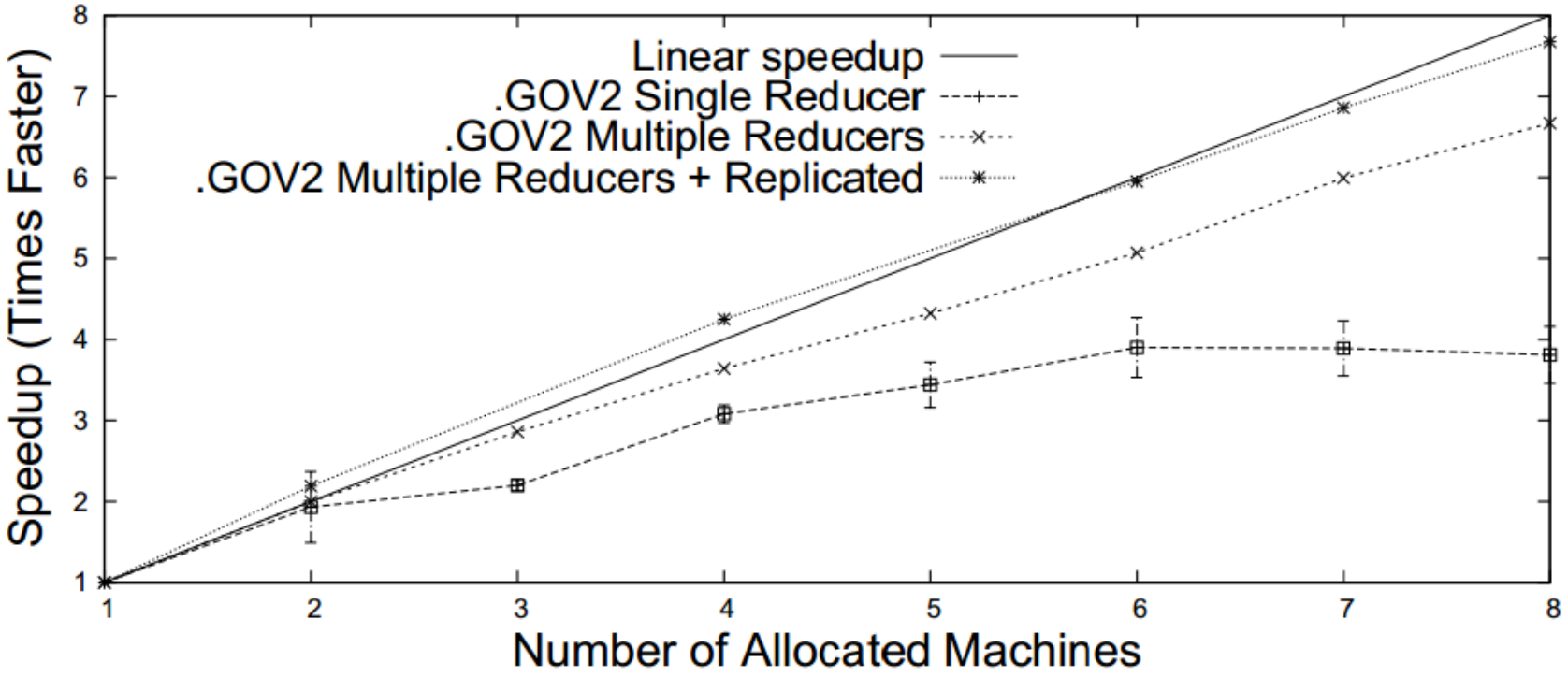
Parametri di MapReduce

- Scelta di numero di mapper e reducer cruciale per l'efficienza
- Numero di reducer generalmente vincolato dal numero backend per le query
- Scelta del numero di mapper non banale

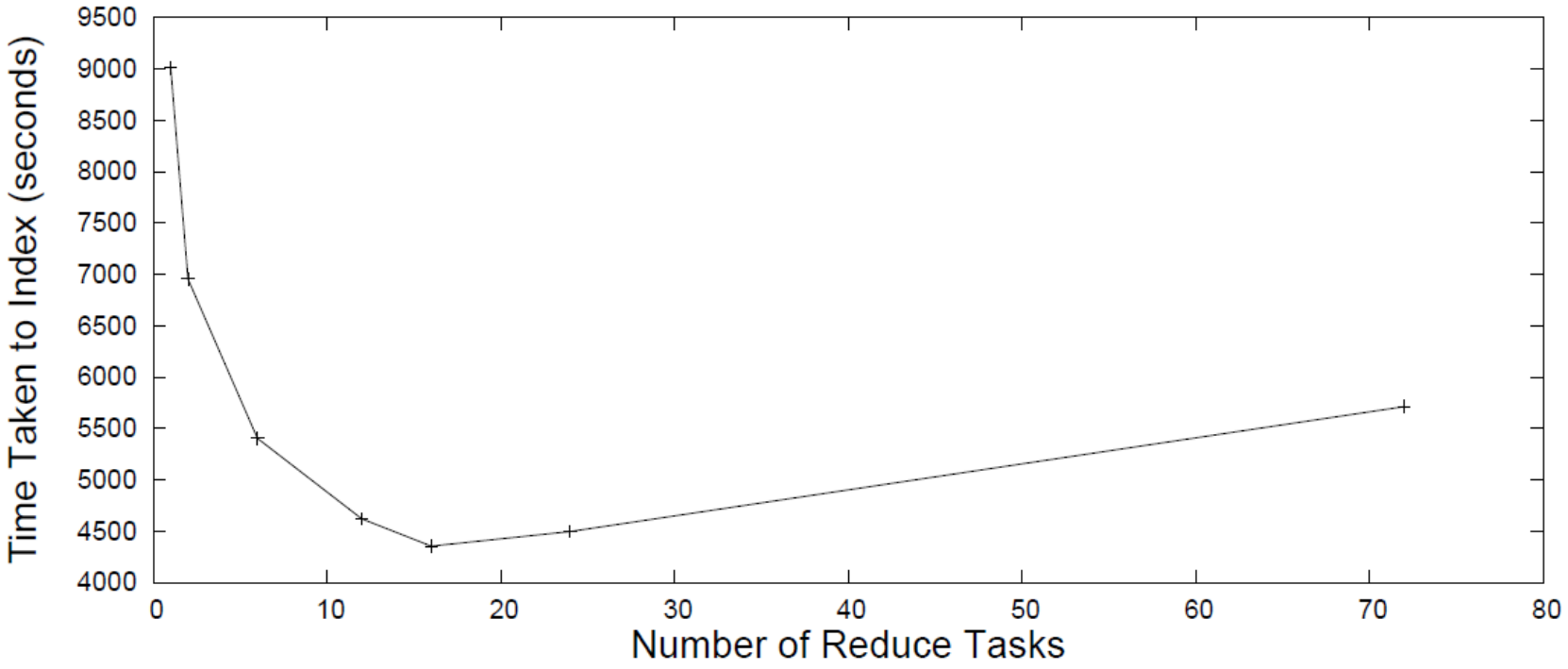
Speedup


Linear speedup
.GOV2 Single-Pass MapReduce Indexing
.GOV2 Dean and Ghemawat
Speedup (Times Faster)
Number of Allocated Machines
1
2
3
4
5
6
7
8

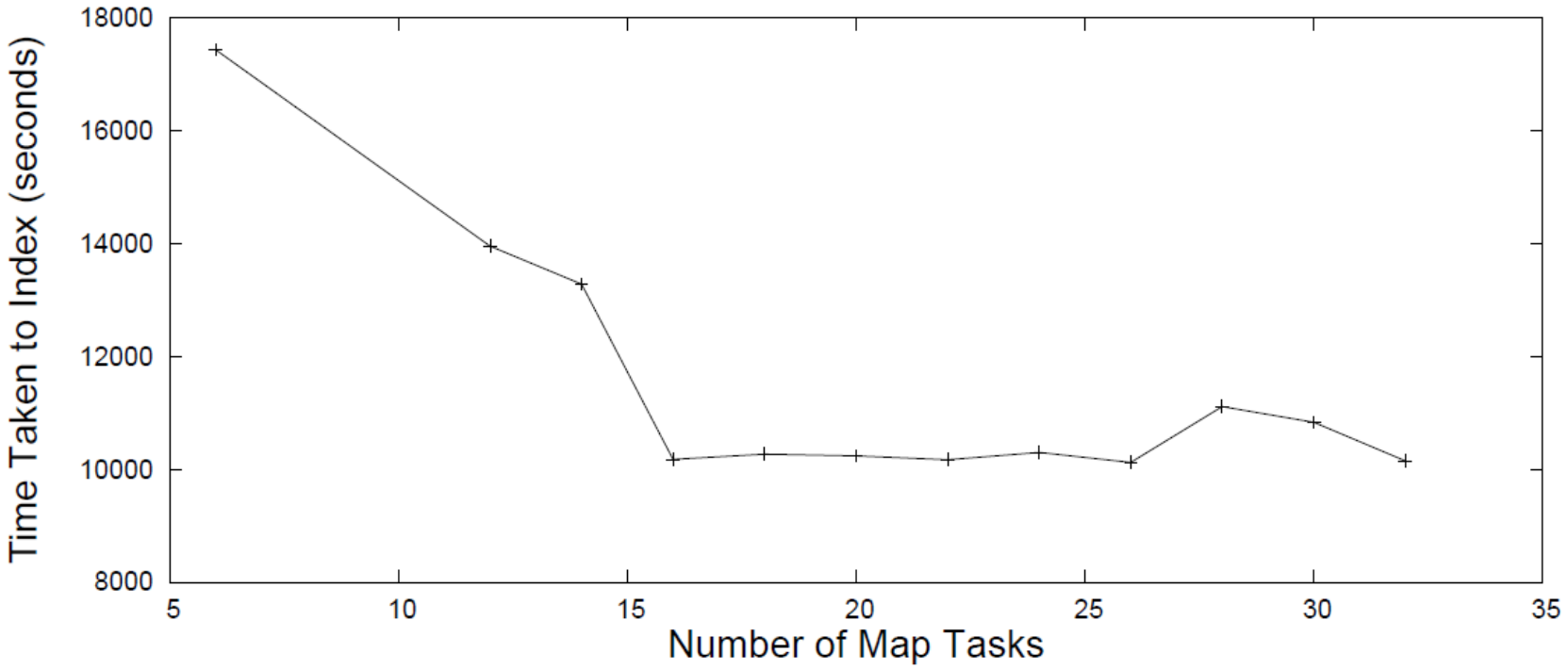
Speedup


Linear speedup
.GOV2 Single Reducer
.GOV2 Multiple Reducers
.GOV2 Multiple Reducers + Replicated
Speedup (Times Faster)
Number of Allocated Machines
1
2
3
4
5
6
7
8

Quanti reducer?



Quanti mapper?



Conclusioni

- MapReduce migliora le prestazioni dell'indexing rispetto al distribuito
- Compromesso tra granularità e carico computativo di mapper e reduce
- Compressione dell'output dei mapper
- Single-Pass MapReduce molto efficiente, ora implementato in Terrier IR

Riferimenti

- On Single-Pass Indexing with MapReduce
  Richard M. C. McCreadie, Craig Macdonald, Iadh Ounis
- Comparing Distributed Indexing: To MapReduce or Not?
  Richard M. C. McCreadie, Craig Macdonald, Iadh Ounis
- Efficient Single-Pass Index Construction for Text Databases
  Steffen Heinz, Justin Zobel
- MapReduce: Simplified Data Processing Large Clusters
  Jeffrey Dean, Senjay Ghemawat
- Bigtable: A Distributed Storage System for Structured Data
  Fay Chang, Jeffrey Dean, Senjay Ghemawat at al.