UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Soluzioni distribuite per la BioInformatica nel Virtual Data Center

Workshop GARR 2017 - Netvolution

Giuseppe Cattaneo

Dipartimento di Informatica
Università di Salerno, I-84084, Fisciano (SA), Italy
cattaneo@unisa.it `http://www.di.unisa.it/~cattaneo`

Roma, 5 Aprile 2017

Overview

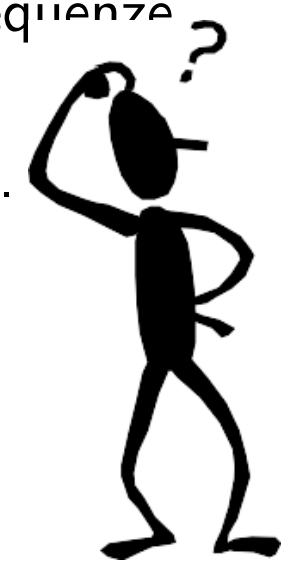
Overview

Biologia e Big Data - (1)

- Next Generation Sequencer (NGS) sono sempre più meno costosi e producono un enorme quantità di dati.
- I dati raccolti richiedono algoritmi sempre più accurati (metagenomica).
- La capacità di calcolo (sequenziale) dei singoli processori ha smesso di crescere da 15 anni.
- L'unica soluzione è lo sviluppo di algoritmi paralleli e *scalabili* per l'analisi di questi dati.

Obiettivi della sperimentazione

- Definire un *benchmark* per avere un riferimento concreto al variare delle configurazione e della tecnologia.
- Raccogliere dati sulle prestazioni raggiungibili sul VDC e sulla effettiva scalabilità
- Individuare una strategia *riutilizzabile* per l'analisi di sequenze genomiche (ovvero Big Data) sul VDC
- Funziona tutto come ci saremmo aspettati ? Ovvero ...



Ovvero:

Ipotesi

È sufficiente utilizzare Hadoop ed un migliaio di core per avere un'applicazione scalabile e performante ?

Risposta

Certamente No !!!

Ovvero:

Ipotesi

È sufficiente utilizzare Hadoop ed un migliaio di core per avere un'applicazione scalabile e performante ?

Risposta

Certamente No !!!

Overview

Il conteggio dei K-meri - (1)

- Sia $\mathcal{S}$ un insieme di stringhe di caratteri dall'alfabeto Σ^*, con $\Sigma = \{A, C, G, T, N\}$.
- **Local Statistics (LS)**

  Quante volte occorre il k-mero sull'alfabeto Σ^k *singolarmente* in ciascuna delle sequenze $\mathcal{S}$.
- **Cumulative Statistics (CS)**
  - Quante volte ciascun k-mero su Σ^k appare *in totale* nelle sequenze $\mathcal{S}$.
- Le sottosequenze di lunghezza k che contengono almeno una occorrenza del carattere N vengono scartate.

Il conteggio dei *K*-meri - (2)

- Le LS vengono utilizzate con il modello *long sequences* e rappresenta un tipico passo di preprocessing delle operazioni di classificazione genome-scale alignment-free.
- Le CS vengono estratte da grandi insiemi di *short sequences*, definite *reads* e sono processate prima dell'operazione di genome assembly.
- Il benchmark sviluppato è stato chiamato *K*-mer Counting on Hadoop (KCH).

Dataset - (1)

- Genoma umano prodotto da (Illumina, 2015).
  - Short sequences (seqs) sull'alfabeto $\Sigma = \{A, C, G, T, N\}$.
- **Dataset per il confronto tra sequenze**:
  - insiemi di *reads* dal DB Illumina.
  - Ogni dataset è memorizzato in un file in formato FASTA.
  - Dataset: ***CS_2GB***, ***CS_8GB***, ***CS_32GB*** e ***CS_128GB***.

```
>SRR034939.189 090406_HWI-EAS68_9096_FC400PR_PE_1_1_10_252 length=100
GGATTGCTAGGTTGAATGGTAGTTCTGTTTTAAGTTCNNNNNNNNNNNTCCNNNNNNNNNNNNNNNNNNNCNNNNNNNANNNNNNNNNNNNCNNNNNNNN
>SRR034939.567979 090406_HWI-EAS68_9096_FC400PR_PE_1_4_1708_1162 length=100
TATAACCTCTTTGGTCCATACACGGTGATGCTGTACCTCACATGTTCCTCAAGCCAACAATGTTCCATTCCTCCTTACATAATAGATCCAGGATGTCATG
>SRR034939.567980 090406_HWI-EAS68_9096_FC400PR_PE_1_4_1708_910 length=100
AGTAATTTTCATTCTATTACATTCCATTCCATTCGAGTACATTGCATTCCGTTCCATTCCATTGCATTCCATTCGAATCCATTCGACTCCACTCCGTTCG
>SRR034939.567981 090406_HWI-EAS68_9096_FC400PR_PE_1_4_1708_828 length=100
TGTAAAGGTGTTTCTCCAAGGCTGGCTGCTGGCTGGAGACAGAAAACTTTTTTGTTTTAAGNTTTTTAGCAAACTCCTTCACAAAGAGATTTCTTTCTGA
>SRR034939.567982 090406_HWI-EAS68_9096_FC400PR_PE_1_4_1708_365 length=100
GGTTCTGAGTGTTTTGAACGAAGACCTTGGAAGCTCTGGGTGGGATTCAGAAATGCTATACNCAGAGCATTCAGATGAGATCTAGAAATGACCTCCTCCT
>SRR034939.567983 090406_HWI-EAS68_9096_FC400PR_PE_1_4_1708_1208 length=100
TCCAGGTNTGTTTCTAACTCCTGCATTACTGATAATATGAAACCTAAACATATAAGCAAAACATATACTACATTCAAGGCAAAGTACATTCTATTACTTT
>SRR034939.567984 090406_HWI-EAS68_9096_FC400PR_PE_1_4_1708_927 length=100
GTGGCTGGTATTGCCCTGTTATCAGGATTTCCTCTGGAGTTTCGGTGTGTGTTTGTTTGATGGCATCTTCGAGGACCCCTTTTCCATGCAGCTGACCCGC
>SRR034939.567985 090406_HWI-EAS68_9096_FC400PR_PE_1_4_1708_788 length=100
TTTACTAAGTTTGCTTTCAAAATGGTCTCATAATTGTAATAGTACATAGAGGTAACATTATCATGCAGCTTTGGTCCATTATTTATATTGGTTATGTTTT
>SRR034939.567986 090406_HWI-EAS68_9096_FC400PR_PE_1_4_1708_1599 length=100
TTCTAAGATAGCTTTAGATTGGTCTTAAATTGAATTTCAAATTATAAAATGATGACCGTAAGTCACTCATGAATTTCTTTAATTCCCCACCTAGATAGAC
```

È sufficiente utilizzare Hadoop ?

Il punto di partenza: `KCH` vs altre implementazioni Hadoop-based.

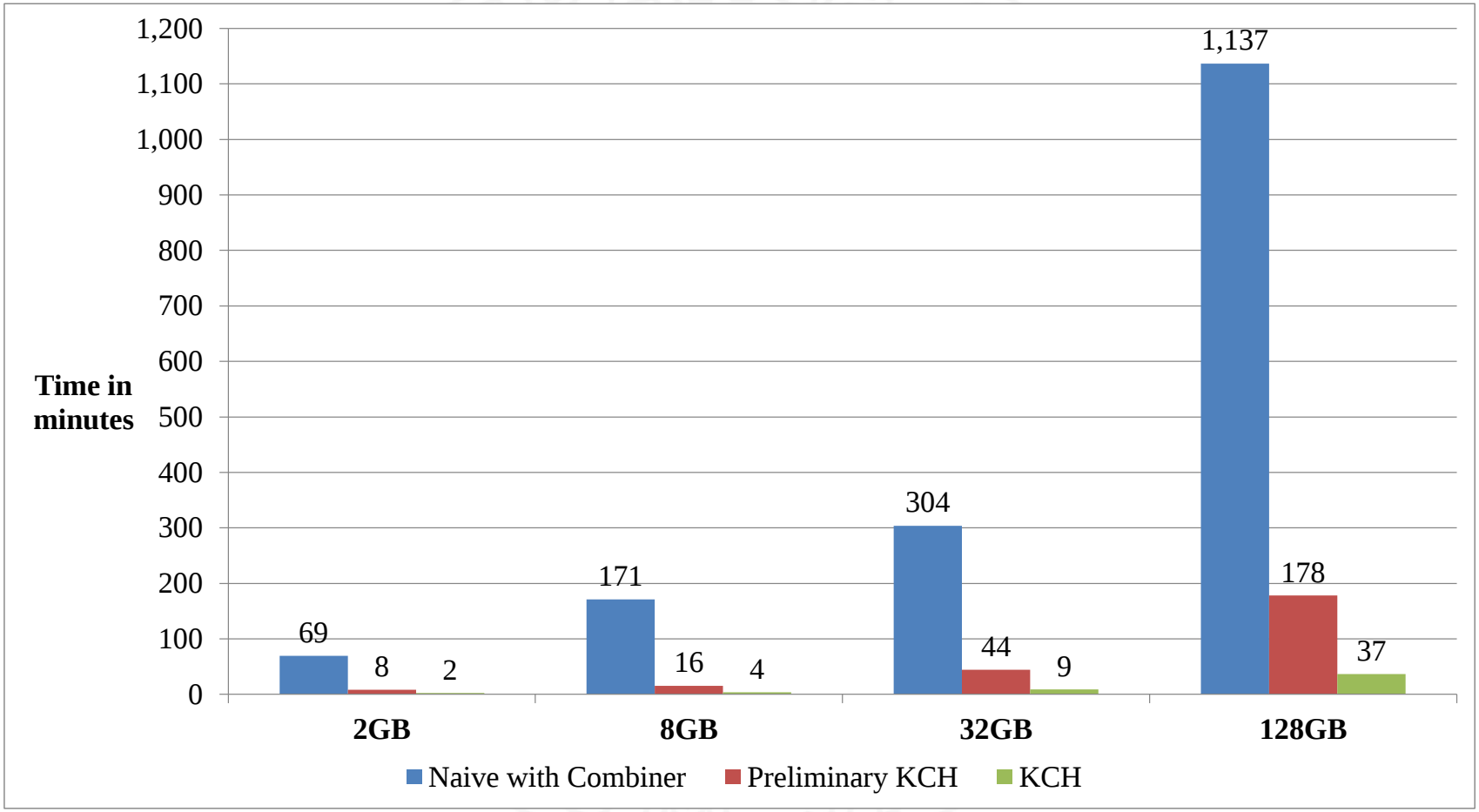
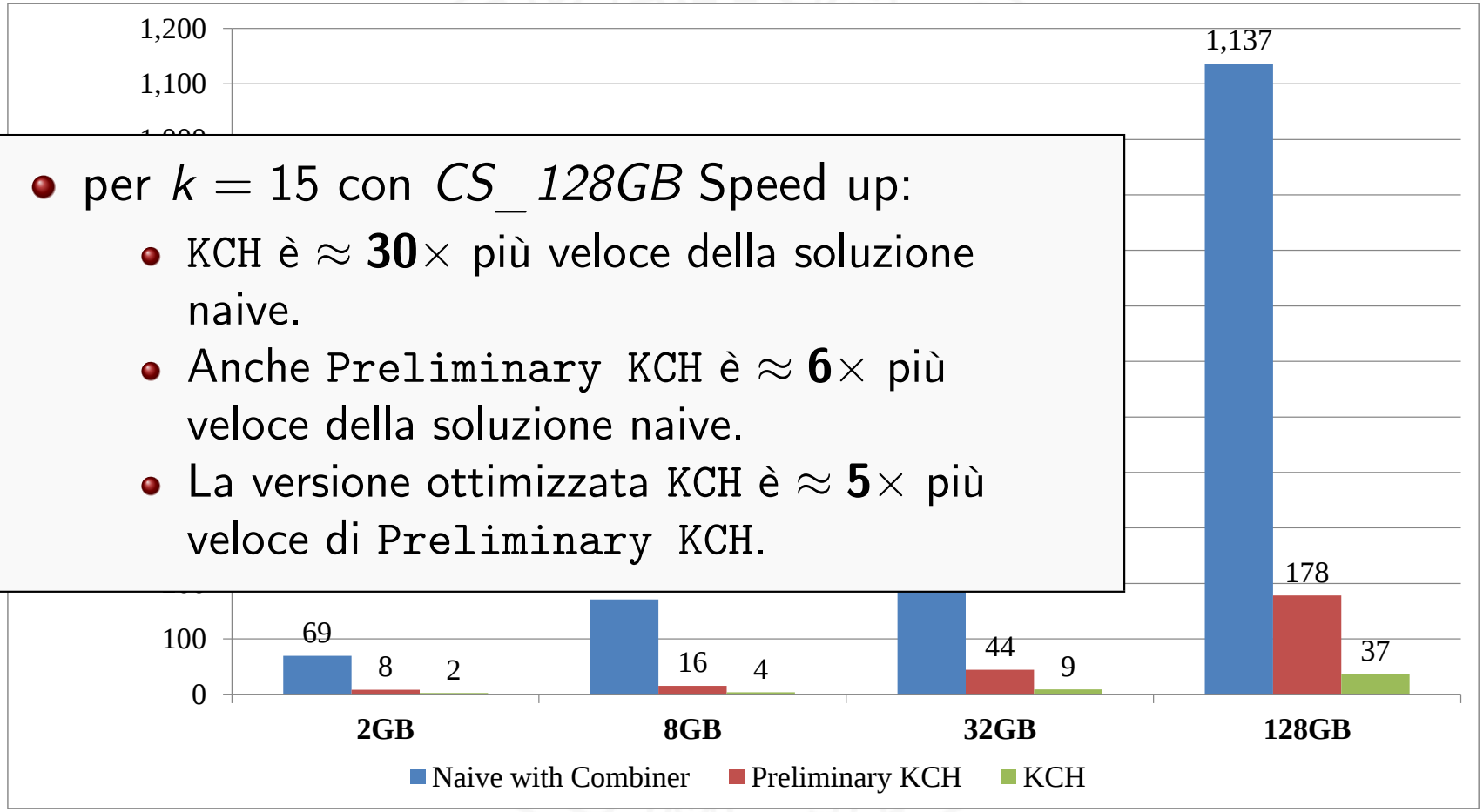


Figura: Confronto tra `KCH` e una prima implementazione *naive* Hadoop-based con uso improprio del Combiner e la versione `Preliminary KCH` con $k = 15$ per ciascun

È sufficiente utilizzare Hadoop ?

Il punto di partenza: `KCH` vs altre implementazioni Hadoop-based.



- per $k = 15$ con *CS_128GB* Speed up:
  - `KCH` è $\approx$ **30**$\times$ più veloce della soluzione naive.
  - Anche `Preliminary KCH` è $\approx$ **6**$\times$ più veloce della soluzione naive.
  - La versione ottimizzata `KCH` è $\approx$ **5**$\times$ più veloce di `Preliminary KCH`.

Figura: Confronto tra `KCH` e una prima implementazione *naive* Hadoop-based con uso improprio del Combiner e la versione `Preliminary KCH` con $k = 15$ per ciascun

KCH versus BioPig - (1)

- BioPig (Nordberg et al., 2013) include un tool per il conteggio dei k-meri.
- È stato utilizzato sullo stesso cluster con 4 slaves e 32 worker.
- Sono stati effettuati test con *CS_32GB* e *CS_128GB* k-meri non canonici con $k = 3$ e $k = 7$.

KCH versus BioPig - (2)

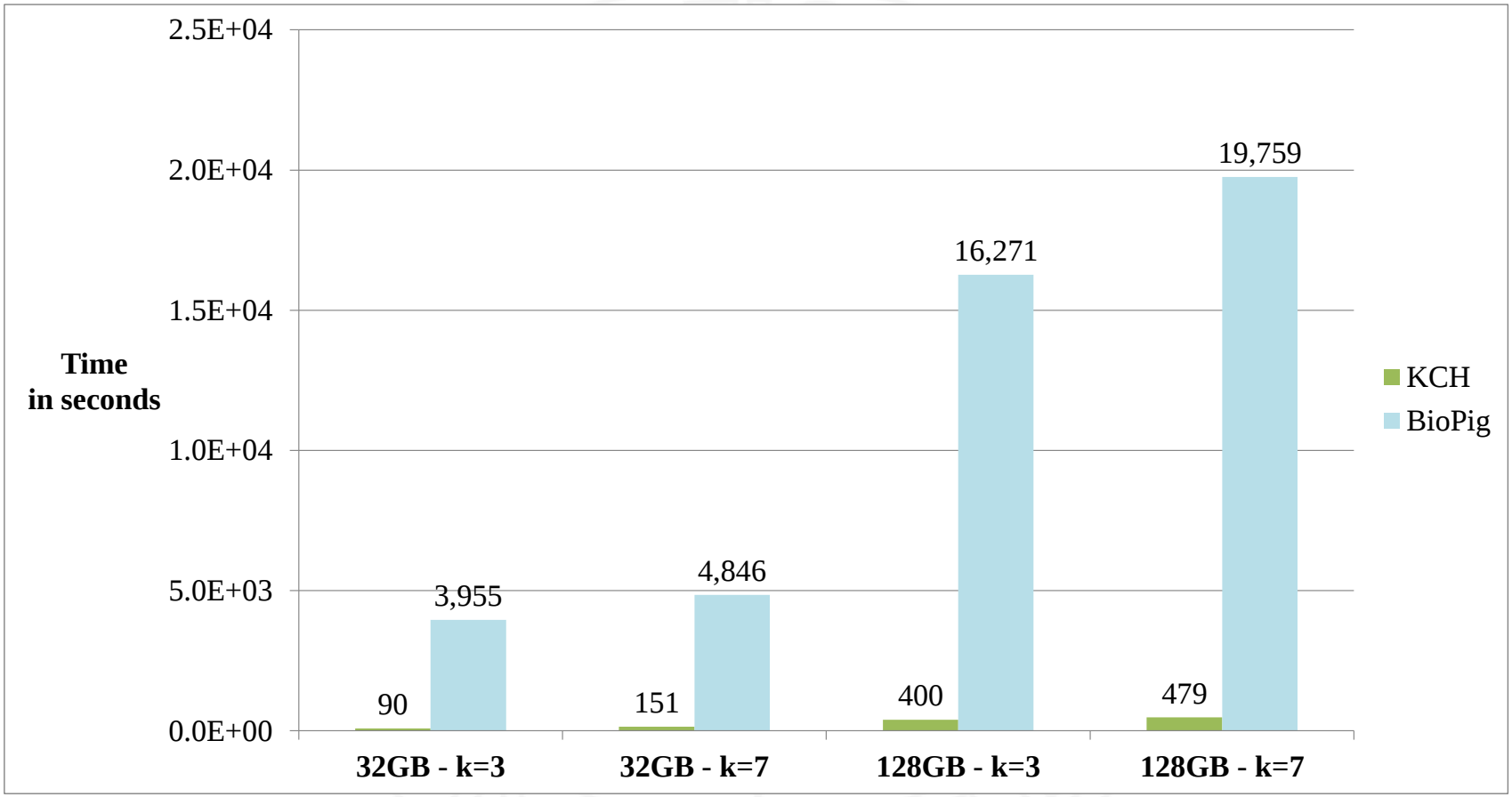


Figura: *Cumulative Statistics*. Results of KCH with respect to BioPig, for the case $k = 3, 7$ (non-canonical k-mers), for 32 GB and 128 GB dataset.

KCH versus BioPig - (2)

- BioPig è davvero lento !!!
- KCH è $\approx$ **40**$\times$ più veloce di BioPig.

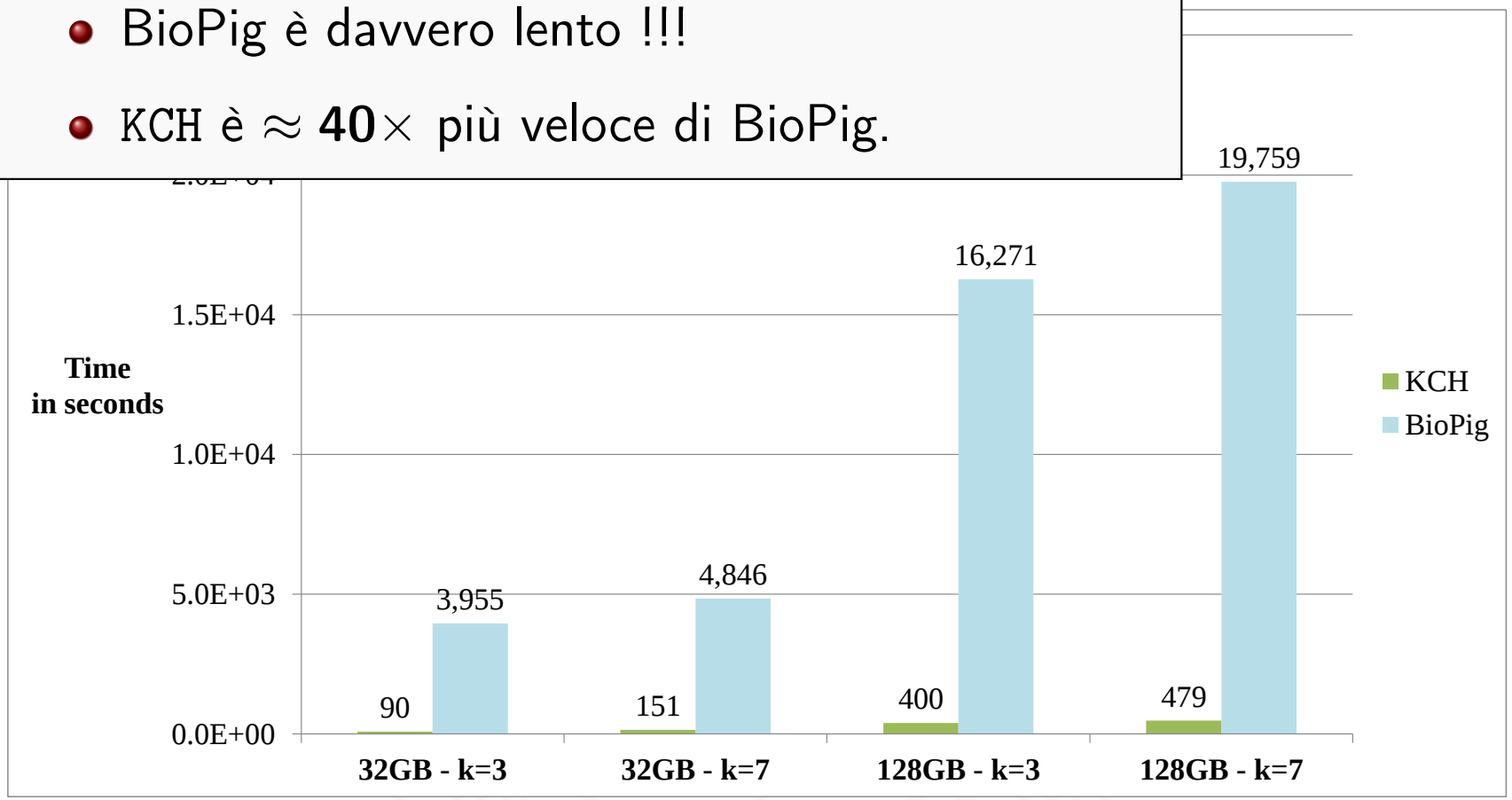


Figura: *Cumulative Statistics*. Results of KCH with respect to BioPig, for the case $k = 3, 7$ (non-canonical k-mers), for 32 GB and 128 GB dataset.

Invece ... KCH sul VDC (1)

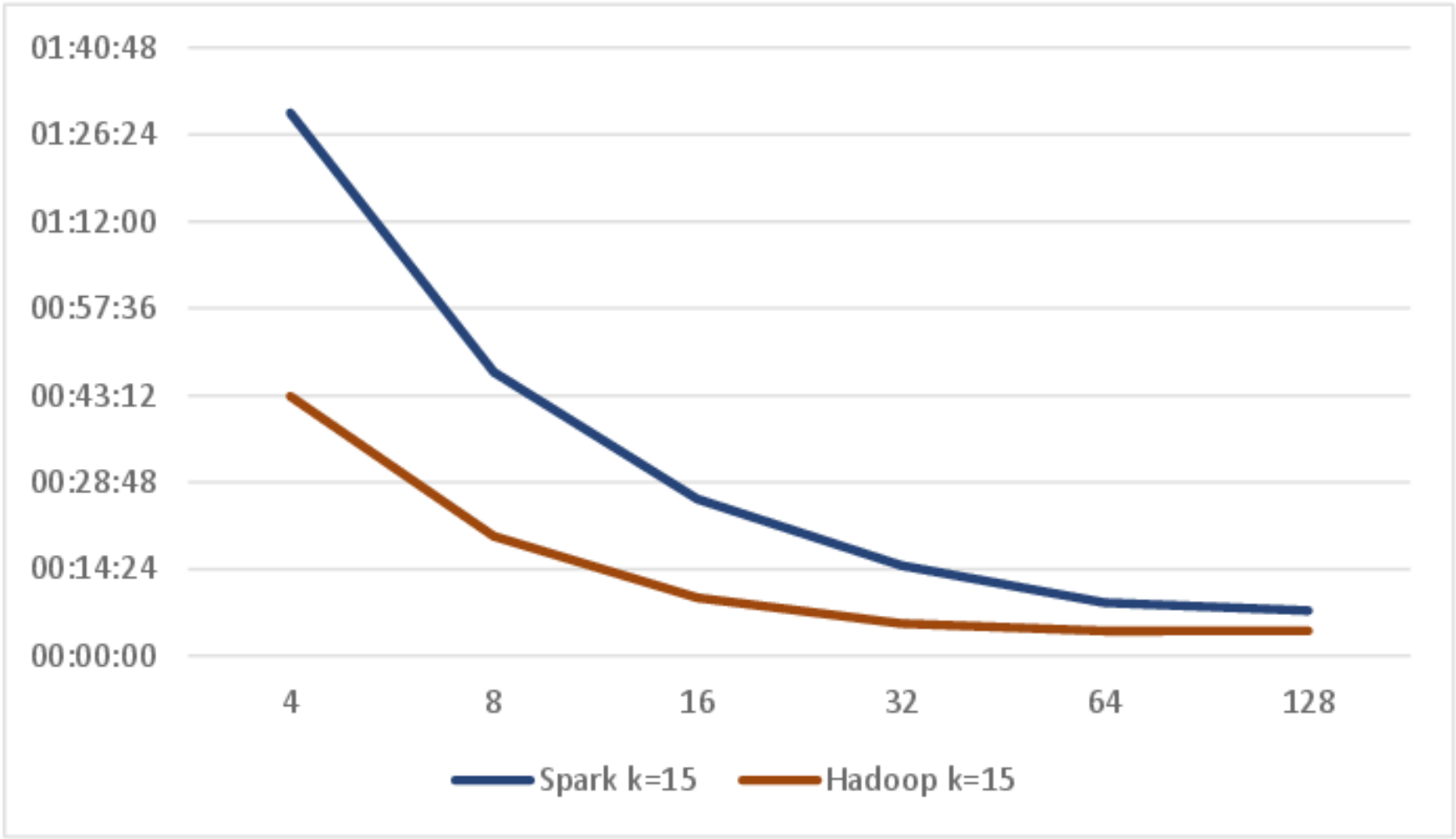


Figura: Tempi di esecuzione di KCH su Spark ed Hadoop per $k = 15$ (k-meri non-canonici), con input di 32 GB.

Invece ... KCH sul VDC (2)

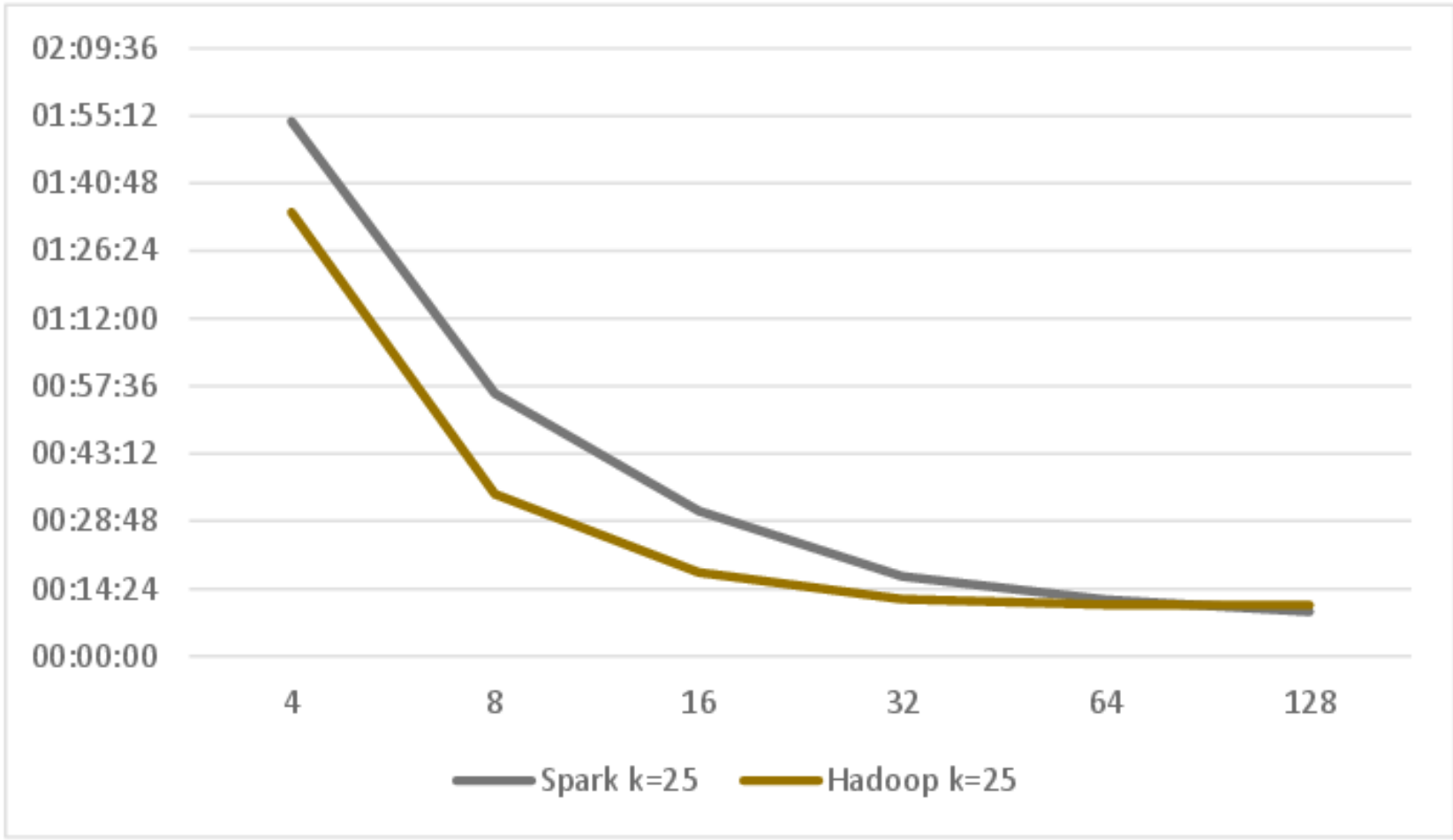


Figura: Tempi di esecuzione di KCH su Spark ed Hadoop per $k = 20$ (k-meri non-canonici), con input di 32 GB.

Invece ... KCH sul VDC (3)

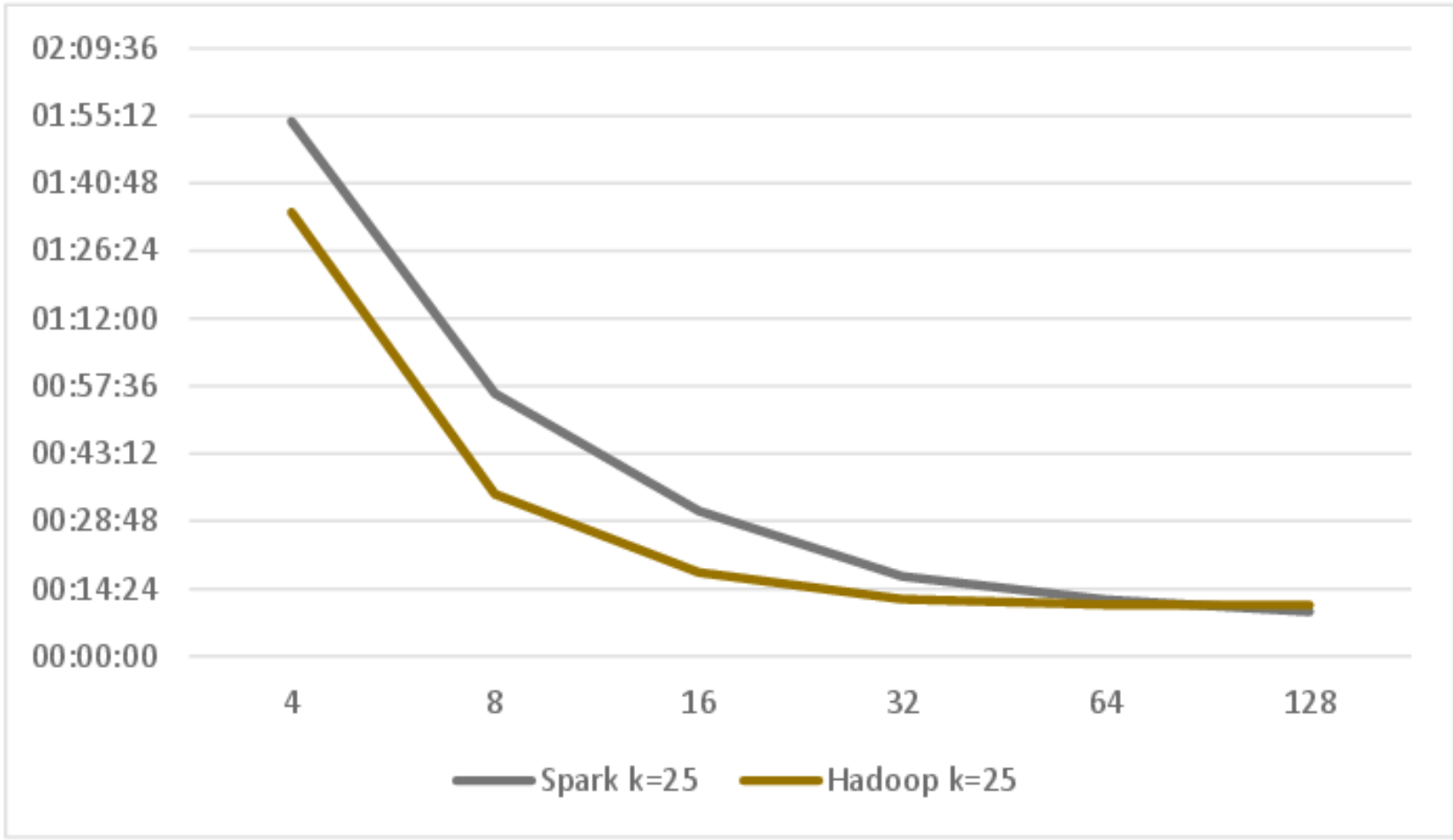


Figura: Tempi di esecuzione di KCH su Spark ed Hadoop per $k = 25$ (k-meri non-canonici), con input di 32 GB.

Invece ... KCH sul VDC (3)

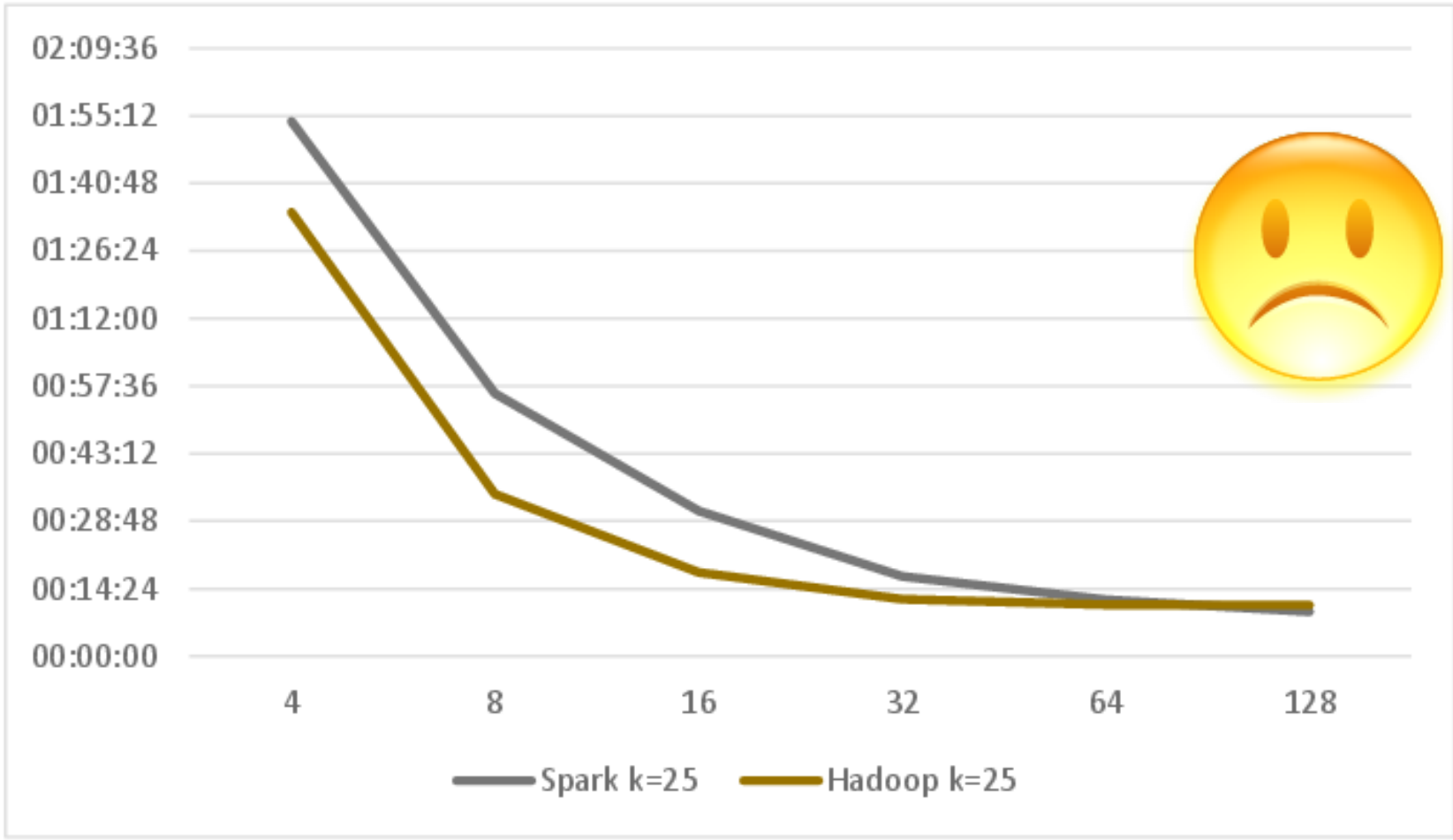


Figura: Tempi di esecuzione di KCH su Spark ed Hadoop per $k = 25$ (k-meri non-canonici), con input di 32 GB.

Overview

L'architettura di Hadoop - (1)

- Lettura di molti dati (possibilmente locali).
- Il **Map Task** estrae dati da ogni record e genera dati intermedi.
- **Shuffle** e **Sort**.
- Il **Reduce** aggrega, somma, filtra o transforma i dati.
- Scrittura dei risultati sull'HDFS.

Architettura HDFS - (1)

Hadoop v2.7.x

- Architettura *master/slaves*.
- Un cluster HDFS consiste di un solo *NameNode*, un *master* server che gestisce il *file system namespace* e regola l'accesso dei client ai file.
- Il *NameNode* può eventualmente essere replicato in un nodo diverso dal master detto *secondary namenode*.
- Ci sono molti *DataNode*, solitamente uno per ogni nodo del cluster.
- I *DataNode* gestiscono lo storage locale dei nodi dove tale servizio è in esecuzione.

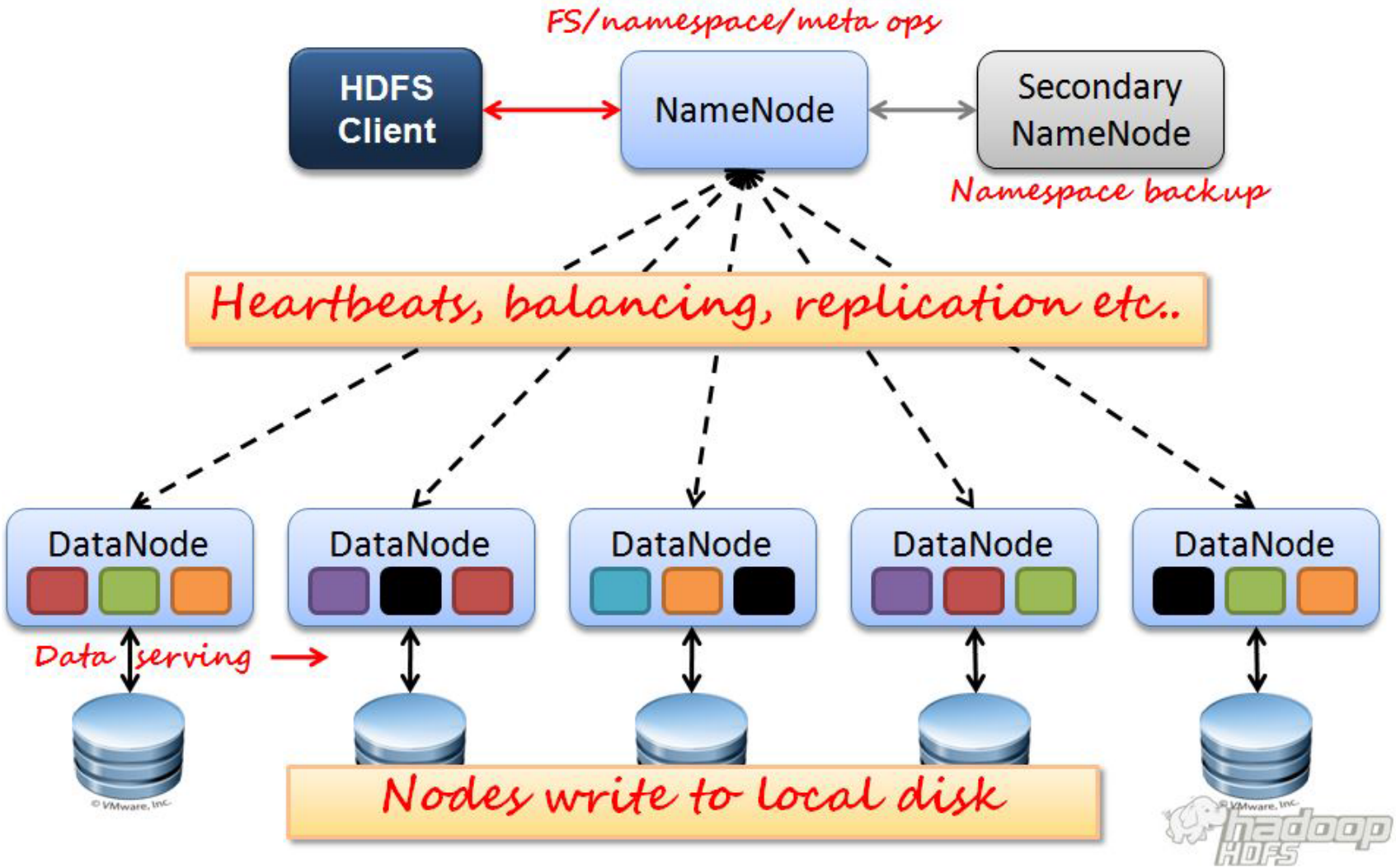
Architettura HDFS - (2)

Hadoop v2.7.x

- HDFS espone un file system namespace e permette di memorizzare i dati in file implementando come richiesto un modello di replica (read-many/write once).
- Un file è diviso in uno o più blocchi che sono memorizzati nei *DataNode*.
- Un *DataNode* serve le richieste di lettura/scrittura; crea, cancella e replica i blocchi su istruzione del *Namenode*.

Architettura HDFS - (3)

Hadoop v2.7.x



Anatomia di un Hadoop Job

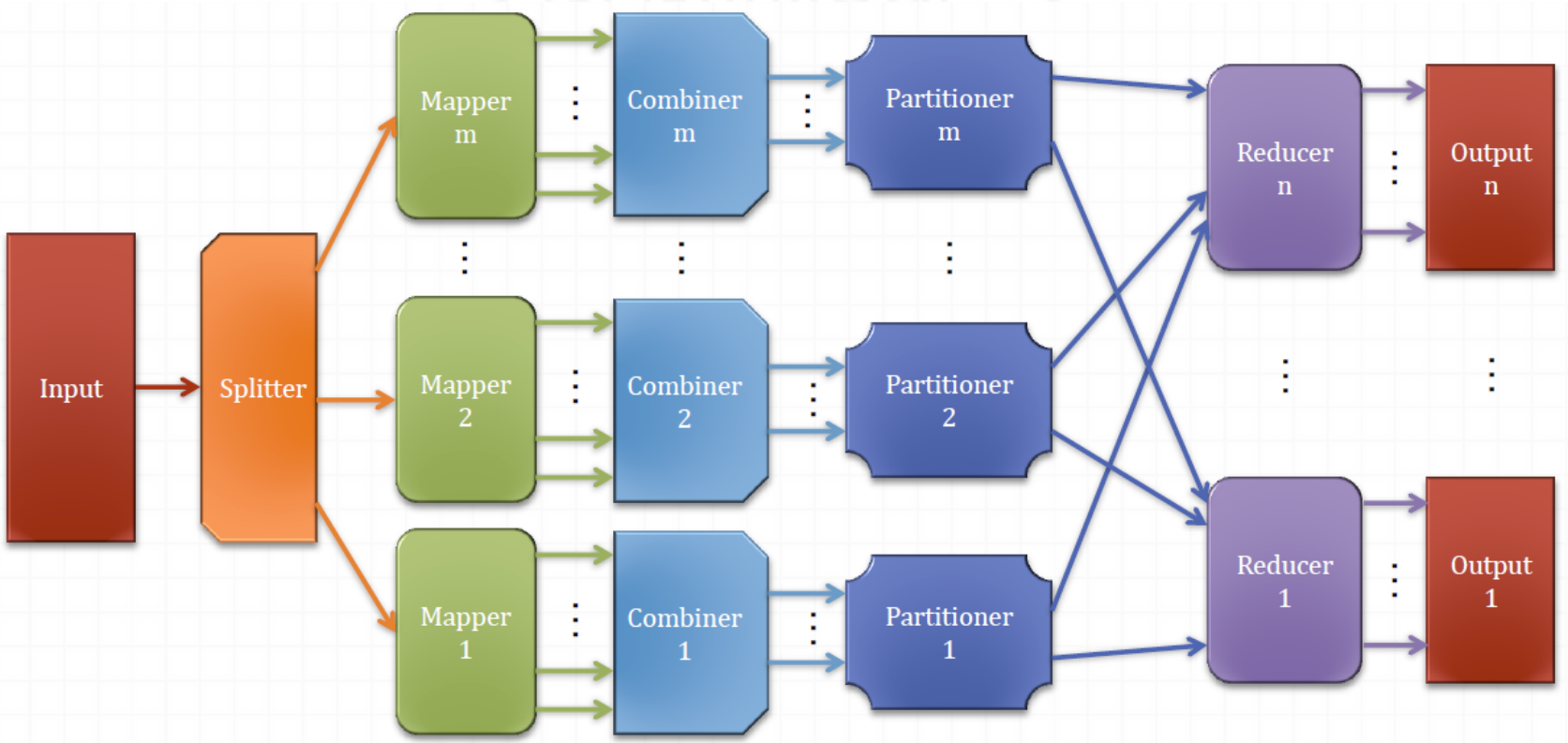


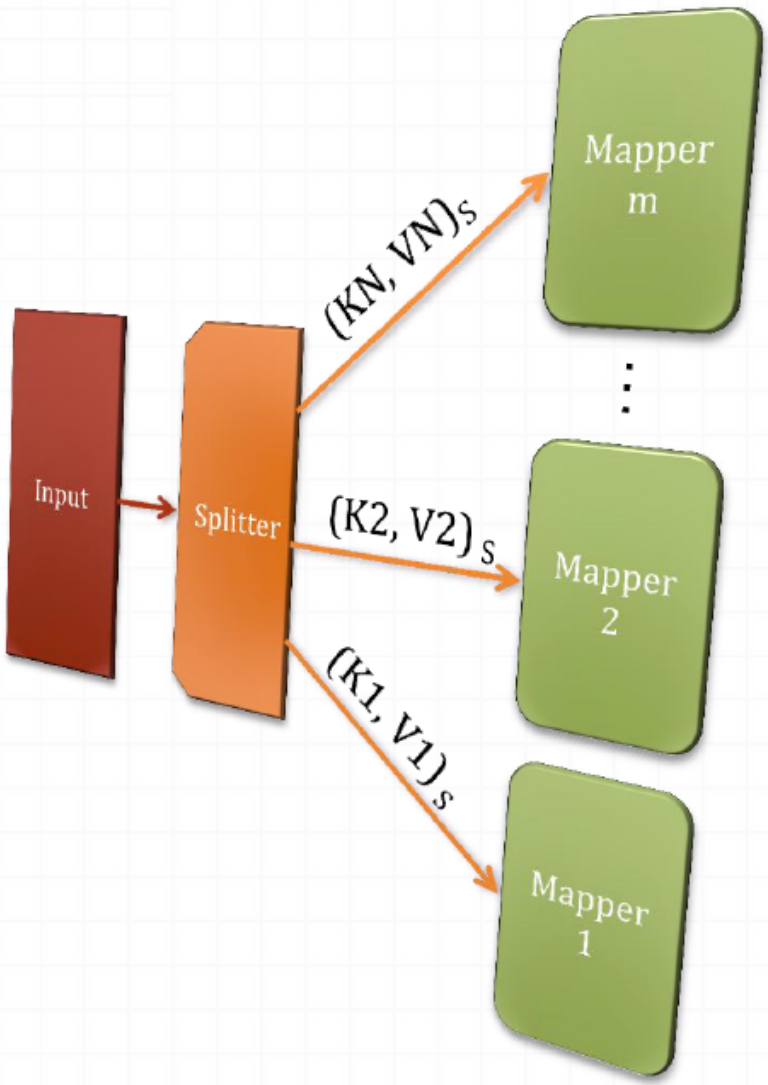
Figura: Anatomia di un programma *MapReduce* in Hadoop.

Records e Data Types

- Un programma *MapReduce*, processa record, cioè una coppia chiave-valore $<K,V>$.
- Sia le chiavi che i valori devono avere una dichiarazione di tipo.
- Esistono anche tipi personalizzabili.
  - Implementare l'interfaccia *Writable*.

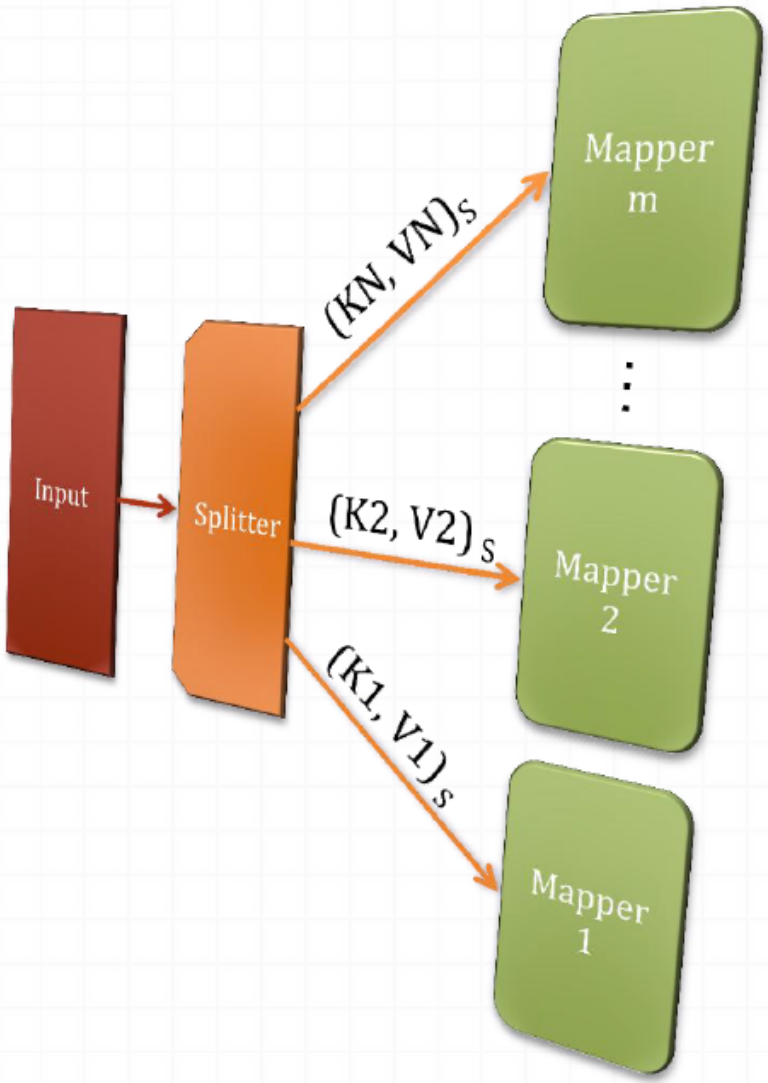
Classe	Descrizione
BooleanWritable	Wrapper alla classe Boolean
ByteWritable	Wrapper per un singolo byte
DoubleWritable	Wrapper per Double
FloatWriable	Wrapper per Float
IntWritable	Wrapper per Integer
LongWritable	Wrapper per Long
Text	Wrapper per testo UTF8
NullWritable	Per chiave o valore nullo

Splitter - (1)



- Lo Splitter determina come dividere l'input in più parti dette *InputSplit*s.
- Gli *InputSplit*s sono processati dai vari *map task* come records $<K,V>$.
- WordCount divide l'input rispetto alle linee: <#linea,contenuto>.

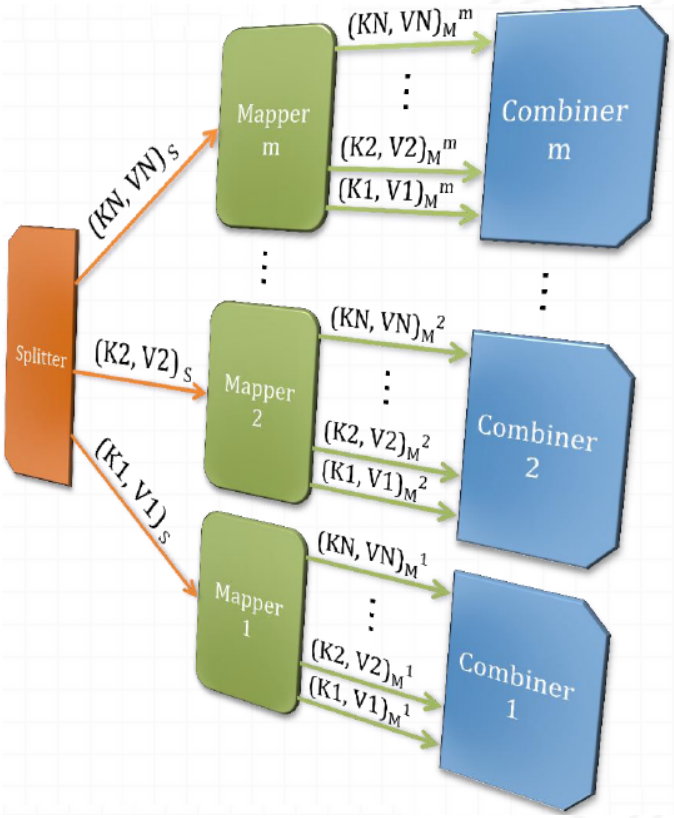
Splitter - (2)



- La chiave ed il relativo valore dipendono dalla classe *InputFormat* implementata.
- Hadoop ne fornisce alcune:

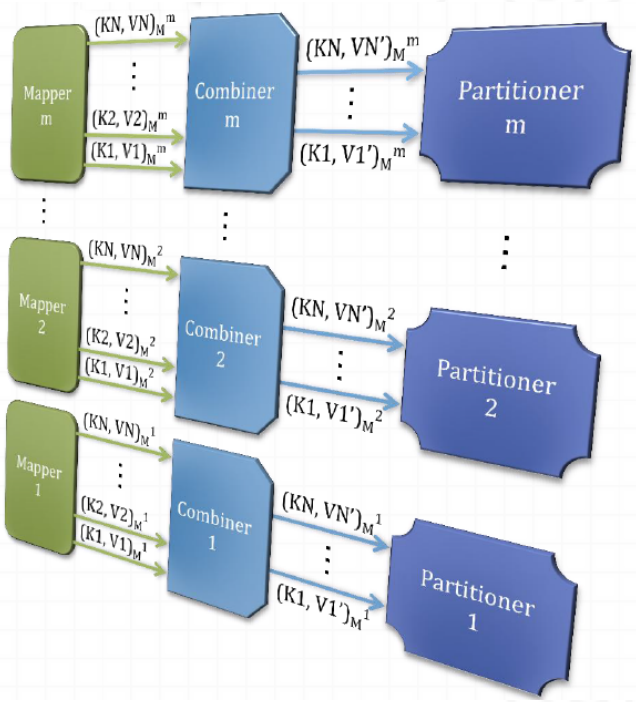
Classe	Descrizione
TextInputFormat	Ogni linea è un record. K1: Offset V1: Una linea di testo
KeyValueTextInputFormat	Ogni linea è un record. K1: Parte prima del separatore (default \t) V1: Parte dopo il sep.
SequenceFileInputFormat<K,V>	Per record binari. K1: input taglia K V1: input taglia V
NLineInputFormat	Ogni linea è un record. K1: Offset V1: Testo

Mapper



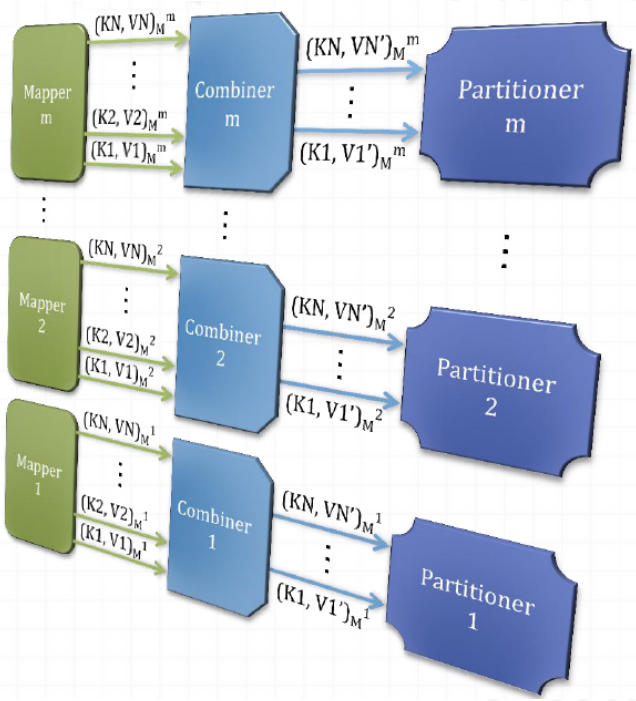
- Processa un record in input restituendo in output una lista di record:
  $<K, V> \rightarrow \text{list}(<K', V'>)$
- In *WordCount*, la linea di testo in input viene divisa in record pari al numero di parole:
  $<\#linea, contenuto> \rightarrow list(<parola, 1>)$

Combiner - (1)



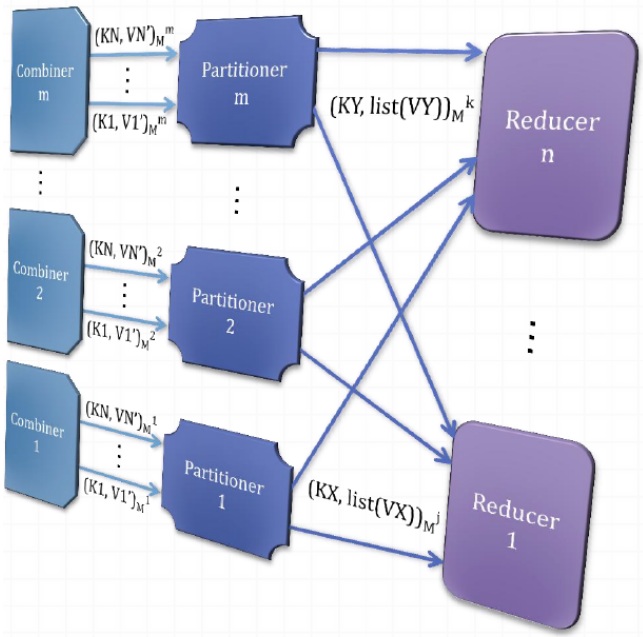
- Componente opzionale.
- Viene detto anche "local reducer" perché si comporta come un reducer
  - ma è eseguito sulla stessa macchina che ha effettuato il *map task*, quindi sfruttando i dati ancora in memoria.
- Applica un algoritmo ad ogni gruppo di record aventi la stessa chiave: $<K, list(V)> \rightarrow list(<K', V'>)$

Combiner - (2)



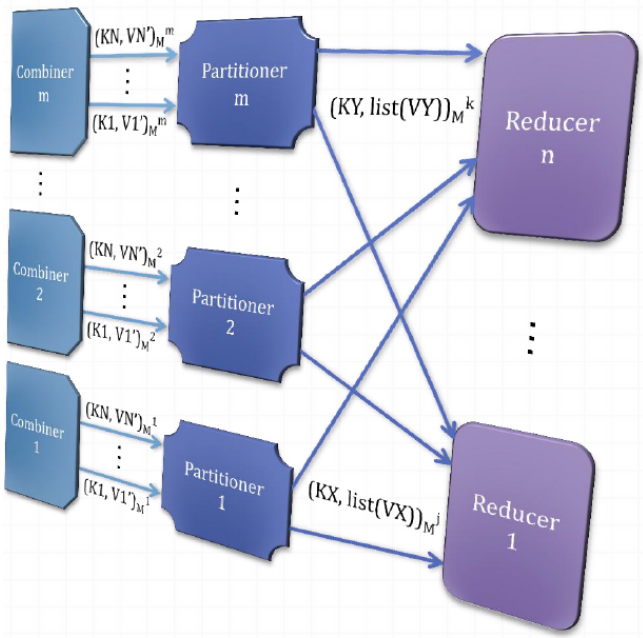
- E' utile per risparmiare network bandwidth.
- In *WordCount*, vengono contate le occorrenze di ogni parola: $<parola, list(1)> \rightarrow <parola, occorrenze>$

Partitioner - (1)



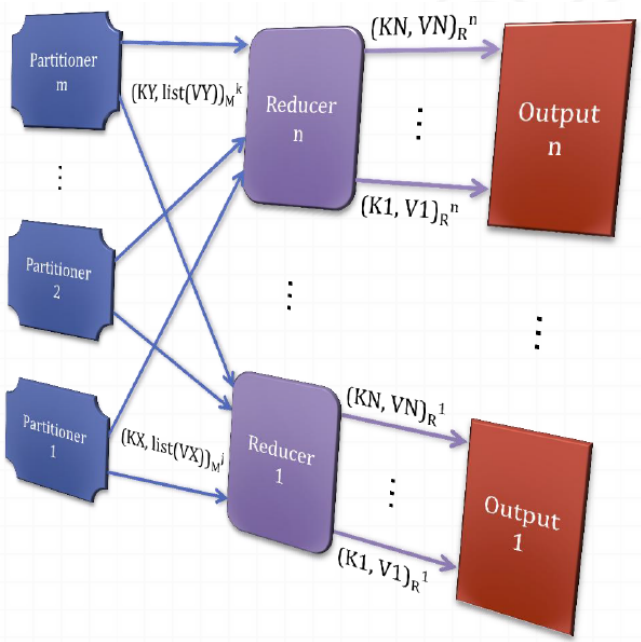
- Come dividere equamente i dati da sottoporre ai reduce?
- Il partitioner è la funzione responsabile della mapping del *key-space* ai reducers.
- Semplice soluzione:
  $hash(key)\%n \rightarrow reduce\ task$,
  n numero di reduce tasks.

Partitioner - (2)



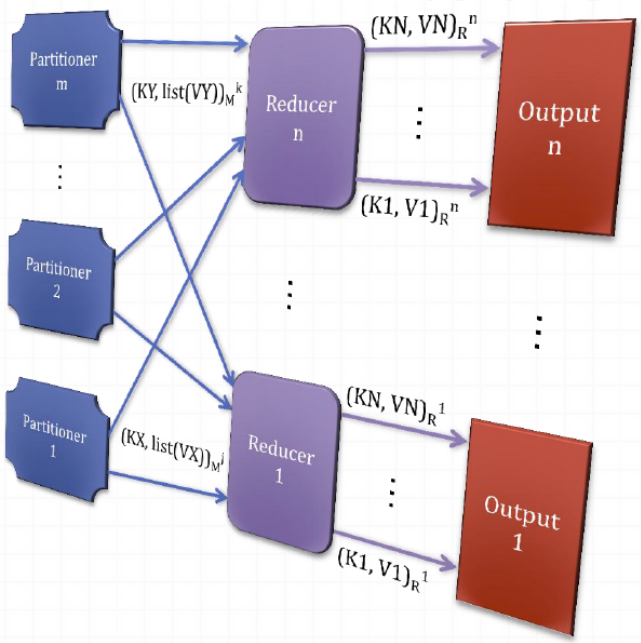
- Il Partitioner stabilisce il criterio con cui i record, in output dal Mapper (oppure, opzionalmente, dal Combiner), raggiungano uno specifico Reducer.
  - Default: *HashPartitioner*, dove il Reducer viene scelto tramite una funzione hash applicata alla chiave.
- *WordCount* sfrutta l'*HashPartitioner*, quindi effettua l'hash sulle parole.

Reducer - (1)



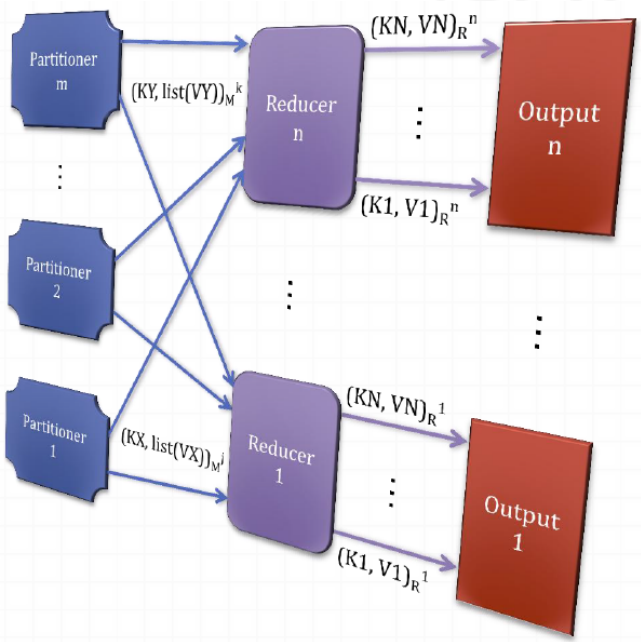
- Il Reducer riceve i record e li ordina in base alla chiave.
- Poi sintetizza gruppi di record aventi la stessa chiave: $<K, list(V)> \rightarrow list(<K', V'>)$.
- I record prodotti dal Reducer vengono restituiti in un file ouput su HDFS.

Reducer - (2)



- In *WordCount*, vengono contate le occorrenze di ogni parola: $<parola, list(occorrenze)> \rightarrow <parola, somma\_occorrenze>$.
- Per ogni Reducer viene prodotto in output un file contenente i record.
- La modalità, con cui vengono stampati i record nei file, viene stabilita dall'implementazione della classe *OutputFormat*.

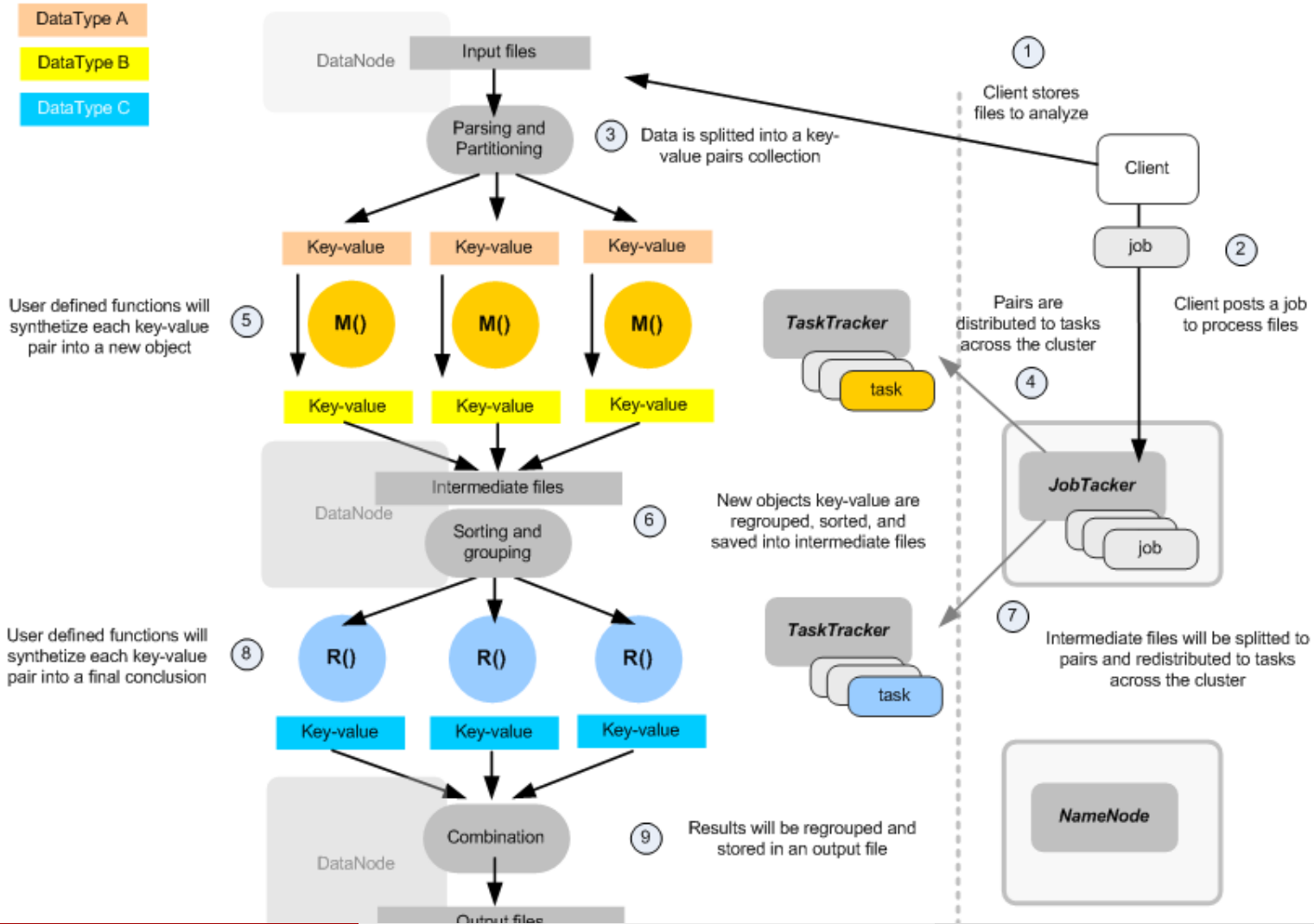
Reducer - (3)



- Hadoop fornisce alcuni *OutputFormat*:

Classe	Descrizione
TextOutputFormat<K, V>	Ogni linea è un record. Chiave e valore vengono separati da un separatore (default \t).
SequenceFileOutputFormat<K,V>	Per il formato binario.
NullOutputFormat<K, V>	Non produce alcun output.

Hadoop *MapReduce* Lifecycle



Overview

Conclusioni

- Non è affatto scontato che applicazioni distribuite possano continuare a scalare quando eseguite su un cluster con storage (SAN)
- La scrittura sull'HDFS è estremamente costosa e lenta. Per la scrittura dell'output è comunque indispenabile utilizzare l'HDFS
  - per l'input split automatico è indispensabile mantenere l'approccio data driven (non basta sostituire l'HDFS con file system distribuiti come lustre)
  - per la raccolta dell'output è comunque indispensabile un file system distribuito tra tutti i nodi slave.
- Il problema è ben noto infatti Spark (RDD) tende a limitare la quantità di dati scritti nella comunicazione tra Map task e Reduce task, ma nel caso del conteggio dei K-meri non è sufficiente (per k grandi).

1

Conclusioni

- Non è affatto scontato che applicazioni distribuite possano continuare a scalare quando eseguite su un cluster con storage (SAN)
- La scrittura sull'HDFS è estremamente costosa e lenta. Per la scrittura dell'output è comunque indispenabile utilizzare l'HDFS
  - per l'input split automatico è indispensabile mantenere l'approccio data driven (non basta sostituire l'HDFS con file system distribuiti come lustre)
  - per la raccolta dell'output è comunque indispensabile un file system distribuito tra tutti i nodi slave.
- Il problema è ben noto infatti Spark (RDD) tende a limitare la quantità di dati scritti nella comunicazione tra Map task e Reduce task, ma nel caso del conteggio dei K-meri non è sufficiente (per k grandi).

1

Conclusioni

- Non è affatto scontato che applicazioni distribuite possano continuare a scalare quando eseguite su un cluster con storage (SAN)
- La scrittura sull'HDFS è estremamente costosa e lenta. Per la scrittura dell'output è comunque indispenabile utilizzare l'HDFS
  - per l'input split automatico è indispensabile mantenere l'approccio data driven (non basta sostituire l'HDFS con file system distribuiti come lustre)
  - per la raccolta dell'output è comunque indispensabile un file system distribuito tra tutti i nodi slave.
- Il problema è ben noto infatti Spark (RDD) tende a limitare la quantità di dati scritti nella comunicazione tra Map task e Reduce task, ma nel caso del conteggio dei K-meri non è sufficiente (per k grandi).

1

Conclusioni

- È in fase di studio una soluzione che sostituisce la comunicazione Map / Reduce con il concetto di *Grid Cache* cf. `https://ignite.apache.org/`
- L'offerta IaaS di GARR potrebbe avere un valore aggiunto se oltre all'infrastrttura potesse offrire *chiavi in mano* la piattaforma (PaaS) e ...
- Un set di applicazioni d'uso generale già pronte per utilizzare al meglio la piattaforma hardware e software (Hadoop).

Conclusioni

- È in fase di studio una soluzione che sostituisce la comunicazione Map / Reduce con il concetto di *Grid Cache* cf. `https://ignite.apache.org/`
- L'offerta IaaS di GARR potrebbe avere un valore aggiunto se oltre all'infrastrttura potesse offrire *chiavi in mano* la piattaforma (PaaS) e ...
- Un set di applicazioni d'uso generale già pronte per utilizzare al meglio la piattaforma hardware e software (Hadoop).

Conclusioni

- È in fase di studio una soluzione che sostituisce la comunicazione Map / Reduce con il concetto di *Grid Cache* cf. `https://ignite.apache.org/`
- L'offerta IaaS di GARR potrebbe avere un valore aggiunto se oltre all'infrastrttura potesse offrire *chiavi in mano* la piattaforma (PaaS) e ...
- Un set di applicazioni d'uso generale già pronte per utilizzare al meglio la piattaforma hardware e software (Hadoop).