Relazione finale del primo progetto del team Takoyaki
Componenti del gruppo : Costa Danilo m. 482492
Frillici Filippo m. 460583

SPECIFICHE PROGETTO

Per quanto riguarda “Una possibile implementazione MapReduce (pseudocodice), Hive e Spark (pseudocodice)”

I file di testo contenenti lo pseudocodice nel formato “Pseudo-Framework-Numero Job” (ad esempio Pseudo-MR-Job1 è la pseudo di MapReduce per il Job 1), stati caricati su repository, oltre che forniti su Moodle, per avere un backup.

Per quanto riguarda “Le prime 10 righe dei risultati dei vari job.” essi sono titolati utilizzando la convenzione: “Output-Framework-Numero Job”.
Sono stati caricati su repository, oltre che scritti qui condensati in 10 righe, per avere una copia di backup sempre disponibile.

Per quanto riguarda “Tabella e grafici che confrontano i tempi di esecuzione in locale e su cluster dei vari job con dimensioni crescenti dell’input”, è tutto nella sezione “tempi esecuzione”.

Per quanto riguarda “Il relativo codice completo MapReduce e Spark (da allegare al documento).” :

Il codice è stato caricato su 2 repo separate github, oltre che caricato su Moodle.

Al seguente link, la repository di Filippo Frillici, è possibile trovare il codice MapReduce e le query Hive, oltre che la pseudocodifica e l’ output:
https://github.com/DarthGalm/progettounobigdata

Al seguente link si trova il codice Spark, la repository di Costa Danilo:
https://github.com/LSparkzwz/spark-big-data-project

Per poter eseguire i job in MapReduce e Hive, operazione preliminare è il caricamento dei dataset in hdfs, come visto a lezione, nella cartella input. Va creata anche una cartella per l' output.

Per compilare il codice Java è possibile semplicemente utilizzare Maven come visto a lezione.

Per comodità sono stati forniti i jar sia della parte Java sia MR che Spark Job 2, chiamato firstproject-1.0.jar, che della parte Spark Job1 e 3, chiamato big-data-project_2.12-0.1.jar entrambi nelle cartelle del progetto.

Per la run del codice Java, da terminale bisogna usare il comando:
"hadoop jar ~/firstproject-1.0.jar NomePackage/NomeClasseMain input/historical_stock_prices.csv input/historical_stocks.csv output/yourfolder".

Per comodità è stato inserito il job Spark in Java direttamente nel progetto dove sono i job in Java MapReduce, per avere un solo package che contenga tutto.

Per la run del Job Spark il comando usato è:
spark-submit --class "JobTwo.SparkEsDue" --master local[*] ~/firstproject-1.0.jar ~/historical_stock_prices.csv ~/historical_stocks.csv
e il risultato è visibile direttamente in terminale

Per la run delle query Hive, da terminale bisogna usare il comando:
"hive -f ~/query.hql"
Ce ne sono quattro, una per creare e popolare le tabelle, le altre una per ogni job.

Compilazione del codice Spark Scala (job 1 e 3):
“sbt clean package” (installare sbt se non lo si ha già)

Run del codice Spark Scala:
“mkdir /tmp/spark-events”
“[percorso di spark]/sbin/start-history-server.sh”
“hdfs dfs -mkdir /resources”
“hdfs dfs -put /resources/historical_stock_prices.csv”
“hdfs dfs -put /resources/historical_stocks.csv”
Per job1:
“spark-submit --class App [percorso del jar]/big-data-project_2.12-0.1.jar 1”
Per job3:
“spark-submit --class App [percorso del jar]/big-data-project_2.12-0.1.jar 3”
NB: se si è compilato il jar esso si trova nella cartell “target/scala-2.12” del progetto.

I risultati sono visibili direttamente nel terminale.

TEMPI ESECUZIONE

Come richiesto vengono riportati in forma tabellare i tempi di esecuzione dei vari job, intesi come tempo che intercorre tra lo start da riga di comando alla effettiva fine del comando, preso anteponendo il comando “time” all’ esecuzione dei comandi dei vari job.

Sono stati utilizzati file via via crescenti, duplicando quelli di input, per ottenere file grandi 2x e 4x rispetto agli originali, con il comando “cat *.csv > fileunico.csv”.

Per le esecuzioni in locale è stata usata sempre la stessa macchina, un MacBook Pro 2018 con 8 GB di RAM e processore 2,3 GHz Intel i5 4-core.
Le versioni sono Hadoop 3.2.1, Hive 3.1.2 e Spark 3.0.0 preview.

Per le esecuzioni in cluster di è stata utilizzata una VM di AWS:
m5.xlarge, 4 vCore, 16 GB RAM, 1 nodo master 2 nodi slave

Nel caso di Hadoop e Hive è stata utilizzata la stessa VM, con versione Hadoop 3.2.1 e Hive 3.1.2
Nel caso di Spark una VM apposita con Spark 2.4.4

Il formato è in minuti:secondi

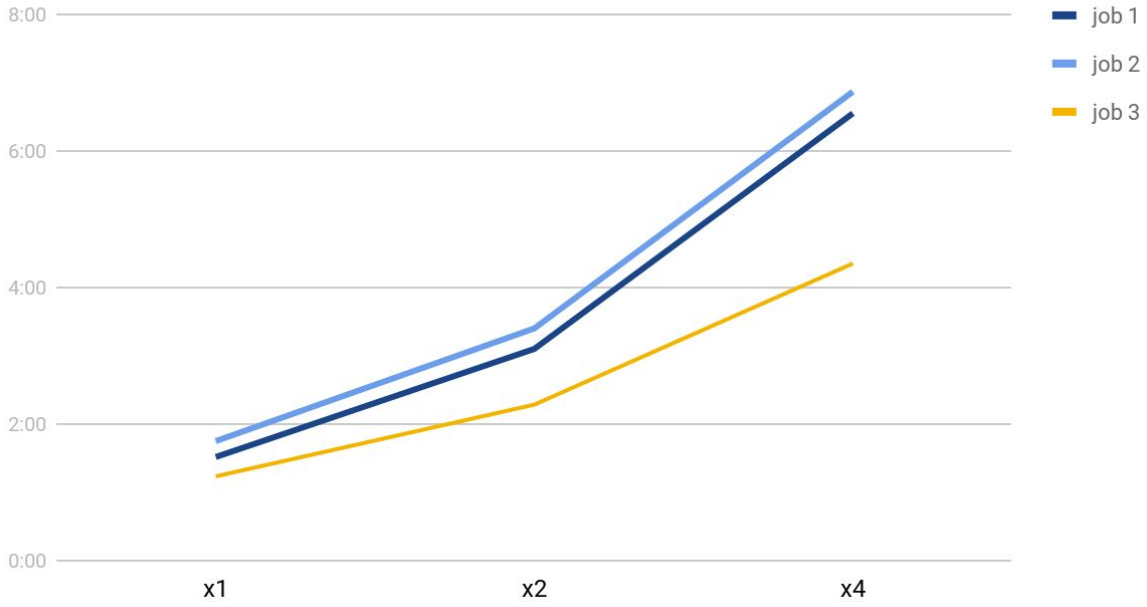
Tempi di esecuzione in locale:

	Job 1	Job 2	Job 3
MapReduce	1x - 1:31 2x - 3:06 4x - 6:33	1x - 1:45 2x - 3:24 4x - 6:52	1x - 1:14 2x - 2:17 4x - 4:21
Hive	1x - 2:01 2x - 4:23 4x - 8:35	1x - 3:06 2x - 7:25 4x - 19:05	1x - 2:03 2x - 3:56 4x - 9:40
Spark	1x - 0:43 2x - 1:12 4x - 2:14	1x - 1:40 2x - 3:12 4x - 7:47	1x - 1:28 2x - 2:11 4x - 3:39

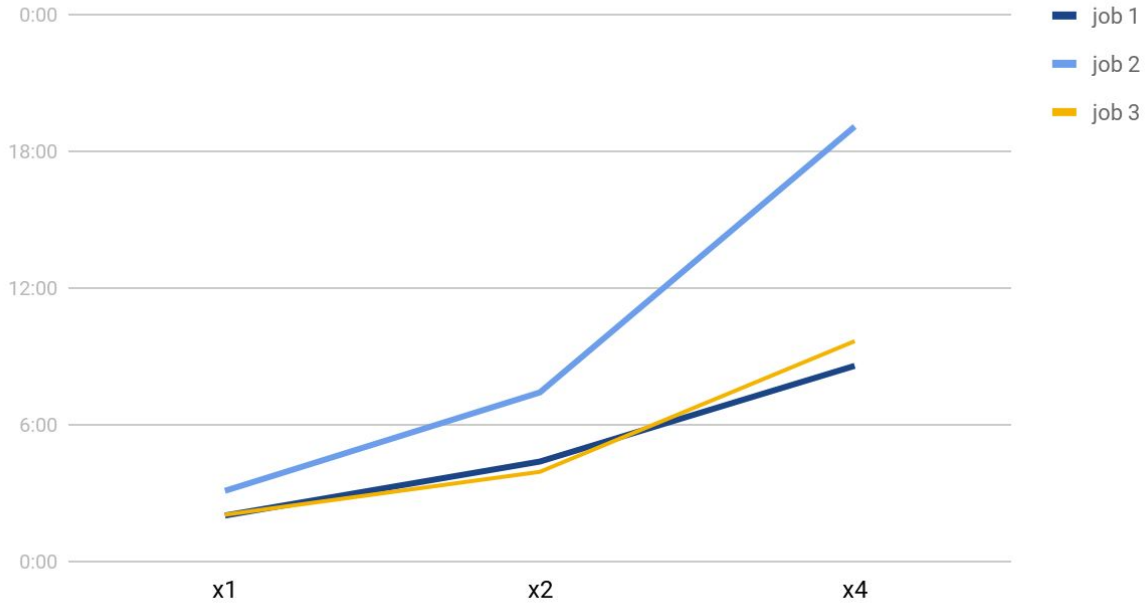
Tempi di esecuzione in cluster:

	Job 1	Job 2	Job 3
MapReduce	1x - 1:12 2x - 1:57 4x - 4:30	1x - 1:34 2x - 2:23 4x - 5:07	1x - 1:39 2x - 2:44 4x - 4:32
Hive	1x - 0:26 2x - 0:43 4x - 1:12	1x - 1:31 2x - 3:00 4x - 8:29	1x - 0:45 2x - 1:09 4x - 1:57
Spark	1x - 0:53 2x - 1:25 4x - 2:30	1x - 1:46 2x - 3:07 4x - 5:58	1x - 1:20 2x - 1:55 4x - 3:12

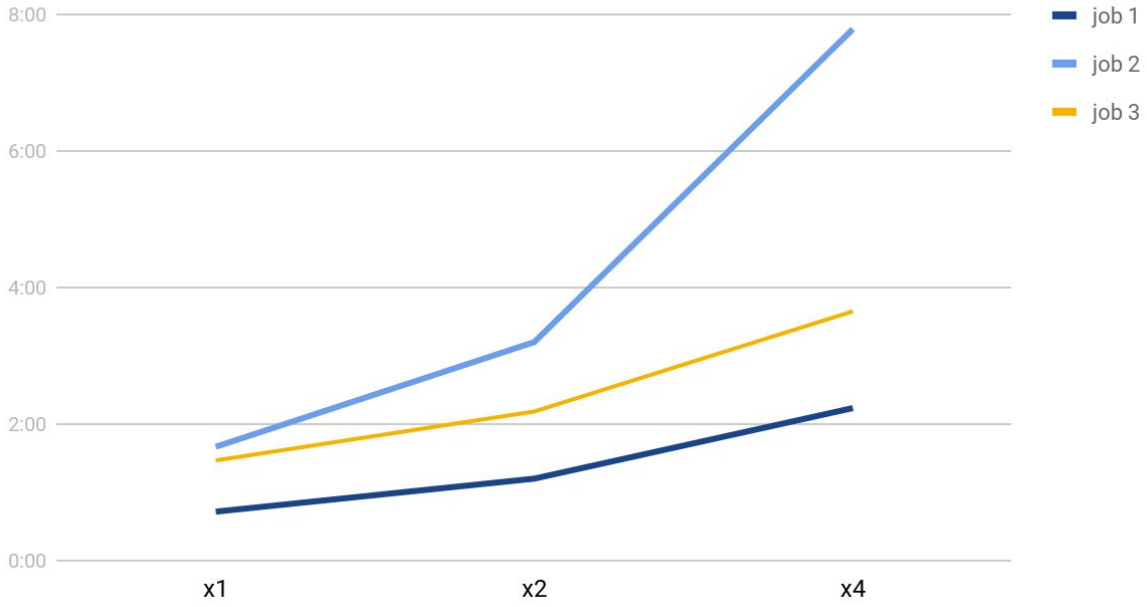
Tempi di esecuzione MapReduce in locale



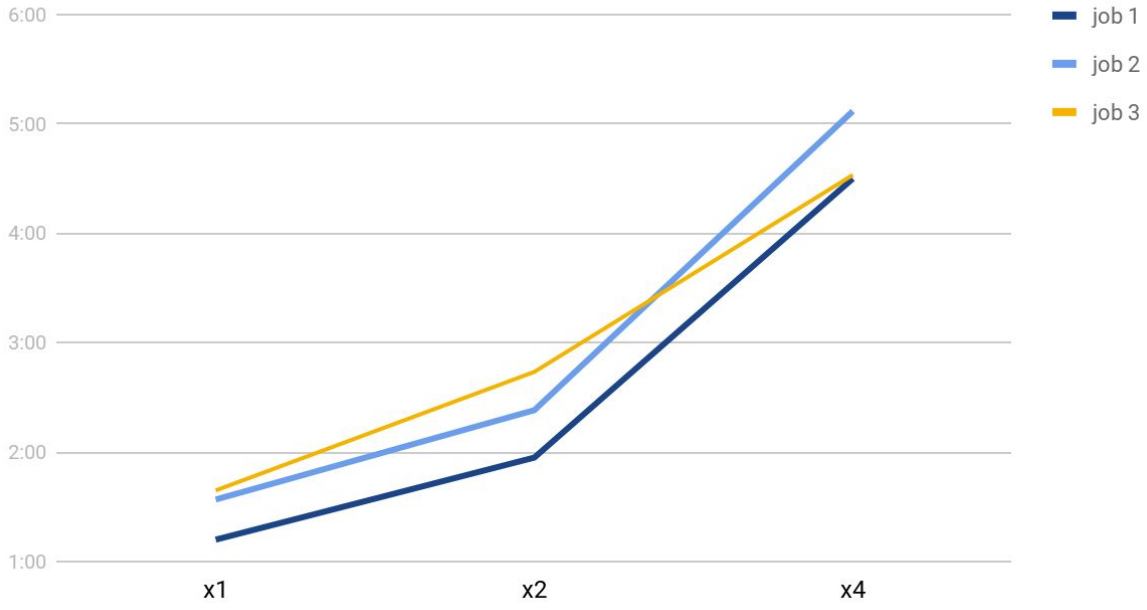
Tempi di esecuzione Hive in locale



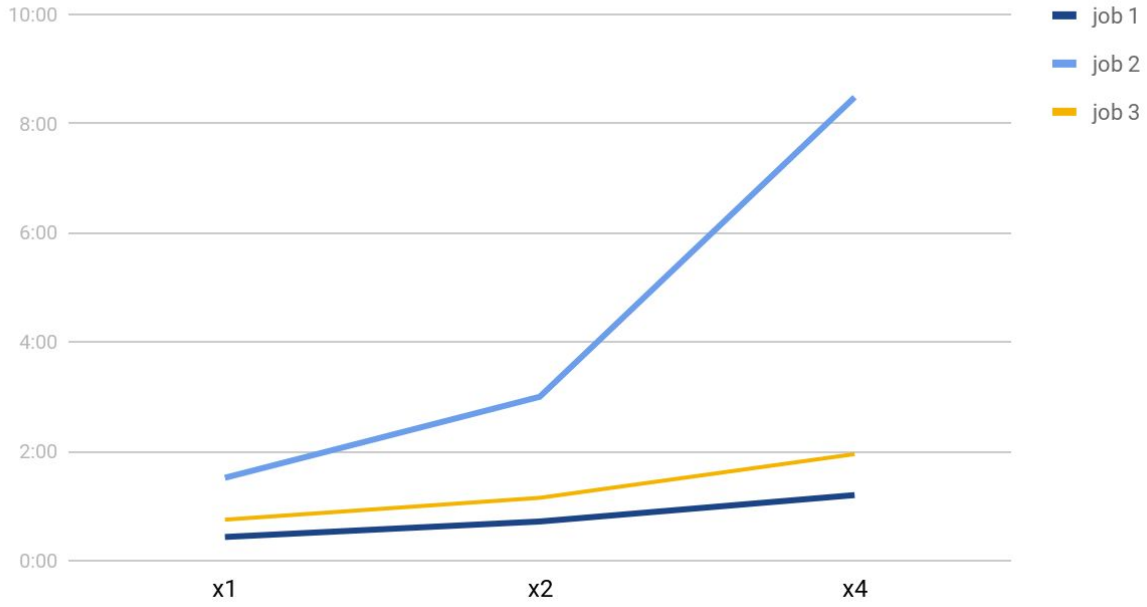
Tempi di esecuzione Spark in locale



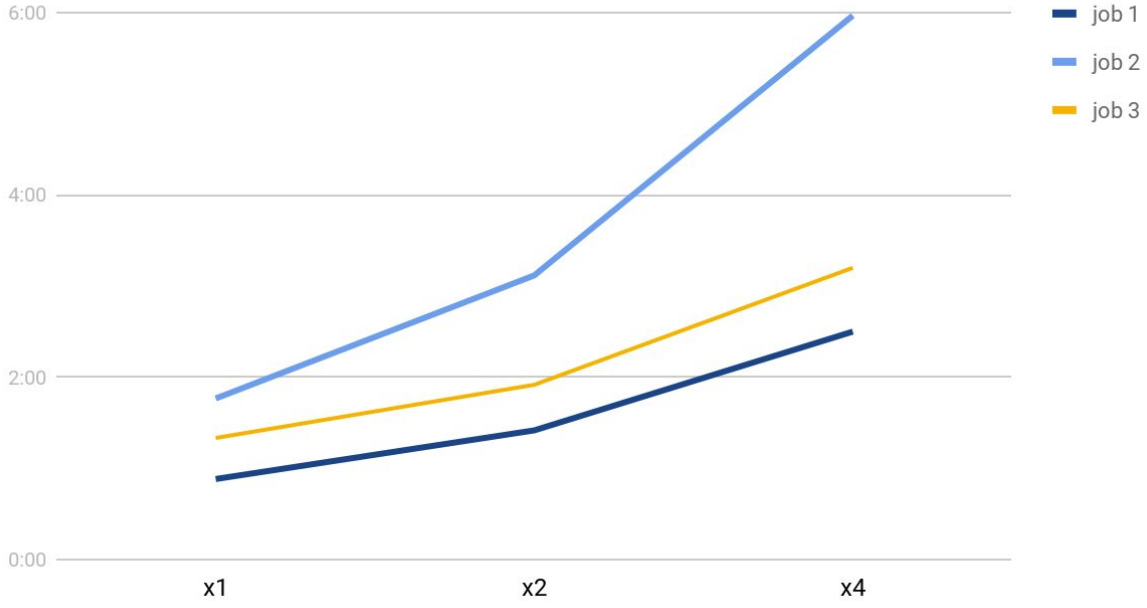
Tempi di esecuzione MapReduce in cluster



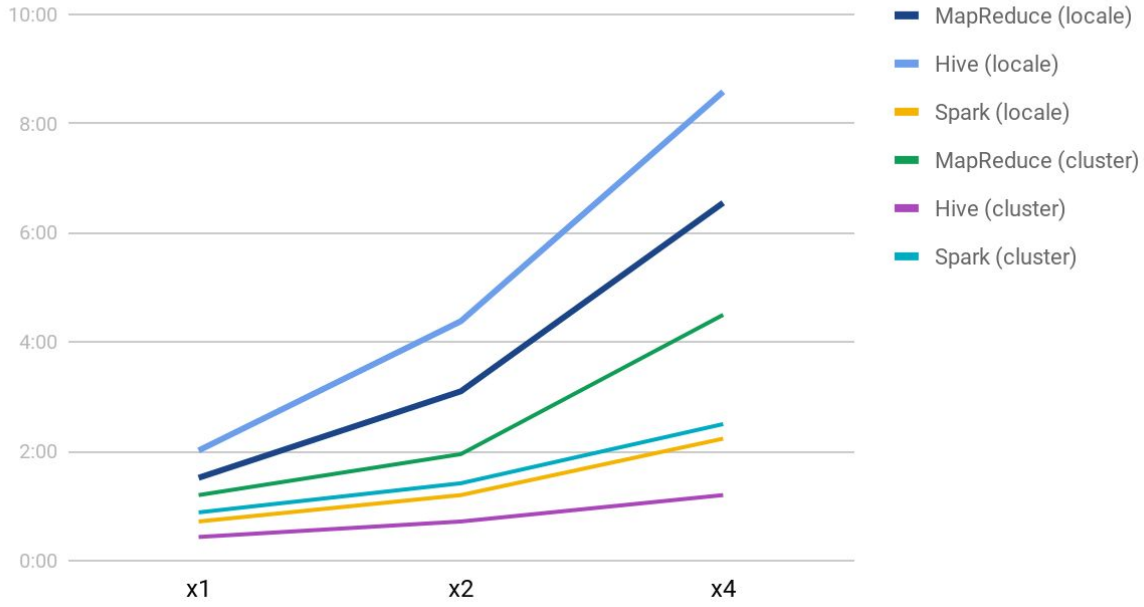
Tempi di esecuzione Hive in cluster



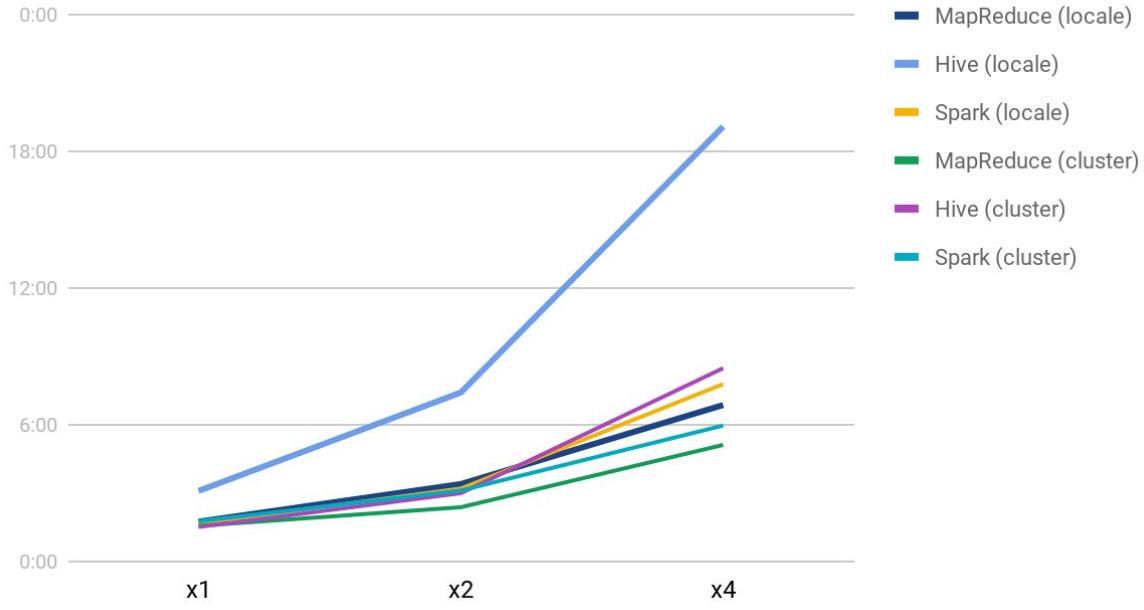
Tempi di esecuzione Spark in cluster

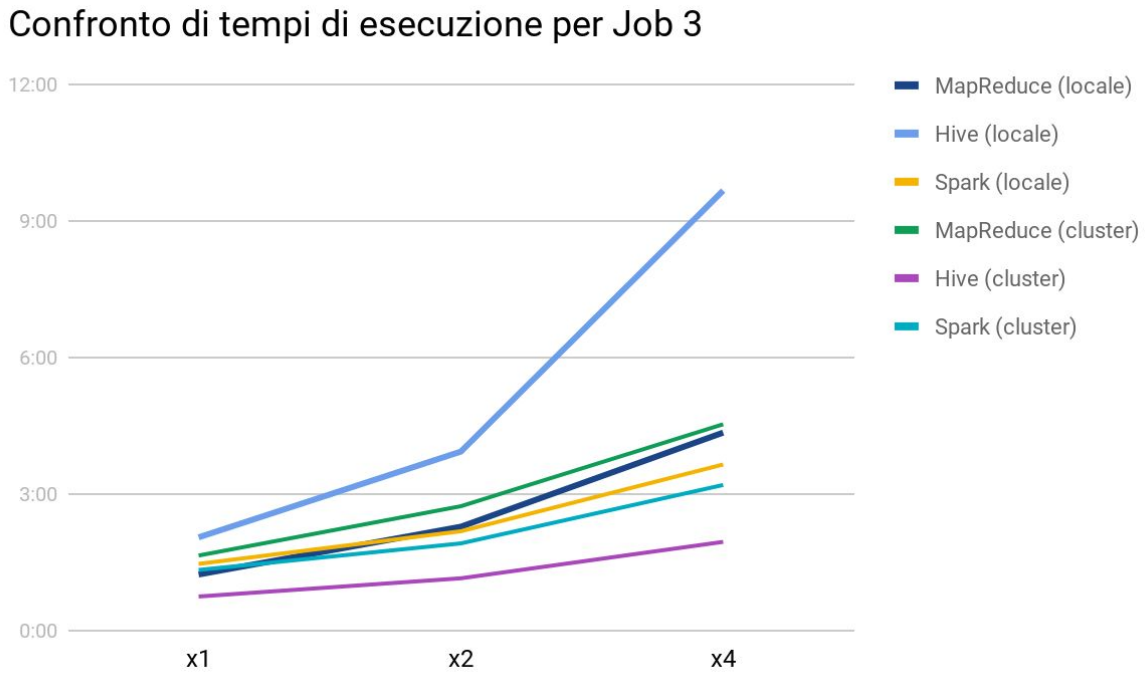


Confronto di tempi di esecuzione per Job 1



Confronto di tempi di esecuzione per Job 2





OUTPUT

È stato richiesto anche di riportare degli output nella relazione, qui di seguito, ricordando che sono disponibili anche online:

Job 1

MapReduce (i valori 0 per arrotondamento a 2 cifre decimali)

EAF percent variation: 267757,13 min: 0,00 max: 22,37 avg: 1245139,04
ORGS percent variation: 217415,93 min: 0,00 max: 19,80 avg: 8570,96
PUB percent variation: 179900,00 min: 0,01 max: 135,00 avg: 34449,40
RMP percent variation: 121081,60 min: 0,03 max: 78,55 avg: 120487,05
CTZ percent variation: 120300,00 min: 0,00 max: 26,73 avg: 52050,27
CCD percent variation: 111250,00 min: 0,01 max: 25,55 avg: 105416,89
SAB percent variation: 110428,77 min: 1,40 max: 38800,00 avg: 1695378,02
KE percent variation: 99400,00 min: 0,01 max: 22,20 avg: 73249,85
LN percent variation: 96700,00 min: 0,01 max: 49,63 avg: 394344,93
GHG percent variation: 64850,00 min: 0,00 max: 24,31 avg: 607659,29

Hive

EAF 267757.0 , 0.002, 22.0, 1245139.0
ORGS 217416.0 , 0.003, 20.0, 8571.0
PUB 179900.0 , 0.009, 135.0, 34449.0
RMP 121082.0 , 0.03, 79.0, 120487.0
CTZ 120300.0 , 0.005, 27.0, 52050.0
CCD 111250.0 , 0.015, 26.0, 105417.0
SAB 110429.0 , 1.398, 318800.0, 1695378.0
KE 99400.0 , 0.015, 22.0, 73250.0
LN 96700.0 , 0.015, 50.0, 394345.0
GHG 64850.0 , 0.005, 24.0, 607659.0

Spark (non arrotondato, per completezza)

(EAF,267757.13458927936,0.0020000000949949,22.3700008392334,1245139)
(ORGS,217415.9261280251,0.00313999992795289,19.7999992370605,8570)
(PUB,179900.0040233133,0.00899999961256981,135.0,34449)
(RMP,121081.60190125568,0.0299999993294477,78.5498504638672,120487)
(CTZ,120300.00230968006,0.00499999988824129,26.7299995422363,52050)
(CCD,111250.00477768485,0.0149999996647239,25.5499992370605,105416)
(SAB,110428.76995446288,1.39849996566772,318800.0,1695378)
(KE,99400.00031664963,0.0149999996647239,22.2000007629395,73249)
(LN,96699.9992052715,0.0149999996647239,49.6300010681152,394344)
(GHG,64850.000307336515,0.00499999988824129,24.3099994659424,607659)

Job2, mostrando solo un settore per confronto

MapReduce:

BASIC INDUSTRIES;{2008: [avgVolume: 2315456, avgPercent: -42,280%, avgClose: 37,656]},{2009: [avgVolume: 2441461, avgPercent: +83,190%, avgClose: 24,876]},{2010: [avgVolume: 2017339, avgPercent: +32,893%, avgClose: 32,350]},{2011: [avgVolume: 1853541, avgPercent: -11,342%, avgClose: 42,742]},{2012: [avgVolume: 1552641, avgPercent: +8,187%, avgClose: 42,373]},{2013: [avgVolume: 1514508, avgPercent: +18,852%, avgClose: 133,946]},{2014: [avgVolume: 1461594, avgPercent: -3,363%, avgClose: 109,492]},{2015: [avgVolume: 1610035, avgPercent: +123,964%, avgClose: 40,486]},{2016: [avgVolume: 1944145, avgPercent: +47,723%, avgClose: 31,494]},{2017: [avgVolume: 1557227, avgPercent: +16,694%, avgClose: 35,556]},{2018: [avgVolume: 1432400, avgPercent: -4,066%, avgClose: 37,616]}

Hive:

I valori vengono simili ma non sempre uguali, è dovuto sicuramente a come Hive gestisce le operazioni di arrotondamento, average etc...

BASIC INDUSTRIES 2008: 2315405.227: -42.052: 39.998, 2009: 2440724.694: 82.323: 25.697, 2010: 2017339.222: 32.893: 32.35, 2011: 1853541.465: -11.342: 42.742, 2012: 1552641.089: 8.187: 42.373, 2013: 1514508.169: 18.852: 133.946, 2014: 1460847.23: -3.349: 109.439, 2015: 1610008.731: 123.461: 40.485, 2016: 1937797.23: 47.536: 31.398, 2017: 1557227.772: 16.694: 35.556, 2018: 1428835.173: -4.051: 37.817

Spark

Anche qui presi risultati di riferimento, non ordinati (questo è dovuto al fatto che Spark richiede di gestire operazioni del genere)

Risultati per il settore : BASIC INDUSTRIES: Per l' anno 2008 : Volume annuale medio: 2315405.23 , Percentuale annuale media: -42.05, Quotazione giornaliera media: 40.00 - Per l' anno 2012 : Volume annuale medio: 1552641.09 , Percentuale annuale media: 8.19, Quotazione giornaliera media: 42.37 - Per l' anno 2018 : Volume annuale medio: 1428835.17 , Percentuale annuale media: -4.05, Quotazione giornaliera media: 37.82 - Per l' anno 2016 : Volume annuale medio: 1937797.23 , Percentuale annuale media: 47.54, Quotazione giornaliera media: 31.40 - Per l' anno 2014 : Volume annuale medio: 1460847.23 , Percentuale annuale media: -3.35, Quotazione giornaliera media: 109.44 - Per l' anno 2009 : Volume annuale medio: 2440724.69 , Percentuale annuale media: 82.32, Quotazione giornaliera media: 25.70 - Per l' anno 2010 : Volume annuale medio: 2017339.22 , Percentuale annuale media: 32.89, Quotazione giornaliera media: 32.35 - Per l' anno 2011 : Volume annuale medio: 1853541.46 , Percentuale annuale media: -11.34, Quotazione giornaliera media: 42.74 - Per l' anno 2015 : Volume annuale medio: 1610008.73 , Percentuale annuale media: 123.46, Quotazione giornaliera media: 40.48 - Per l' anno 2013 : Volume annuale medio: 1514508.17 , Percentuale annuale media: 18.85, Quotazione giornaliera media: 133.95 - Per l' anno 2017 : Volume annuale medio: 1557227.77 , Percentuale annuale media: 16.69, Quotazione giornaliera media: 35.56

Job 3:

Questo è problematico non a livello concettuale, ma a livello di output. Essendo stato fatto in maniera differente, gli output cambiano.

Questo è stato fatto poiché eravamo consapevoli dei problemi che poteva dare il Job 3 (dato che senza arrotondamento in prima battuta, non vi erano trend in comune) e abbiamo deciso di approcciare il problema in maniera differente per vedere come sarebbero cambiate le cose.

Ci sono delle corrispondenze tra trend di job diversi, ma in generale tutti e 3 i job sono stati gestiti in maniera differente (spesso portando a minime differenze dal punto di vista di valori, di 1 punto percentuale, ma enormi dal punto di vista del raggruppamento!)

Ad esempio in Hive è stato usato il metodo ROUND, in Java è stato utilizzato un semplice cast da double a long (e quindi è stato lasciato a Java il compito di arrotondare) mentre in scala anche lì, il metodo round().

Inoltre l'arrotondamento è stato fatto in punti differenti del job, (ad esempio in MR viene fatto al momento di calcolare la media, in Hive dopo il calcolo della media, quando è ora di raggruppare e stampare il risultato, mentre in Scala si è arrotondato sia prima di sommare i valori, che dopo aver effettuato la media.

MapReduce:

{LAKELAND INDUSTRIES INC., GAP INC. (THE)}: 2016:-12%, 2017:45%, 2018:-12%
{ISHARES MBS ETF, ISHARES 7-10 YEAR TREASURY BOND ETF}: 2016:-1%, 2017:0%, 2018:-2%
{FIRST TRUST INTERNATIONAL MULTI-ASSET DIVERSIFIED INCOME INDEX, FIRST TRUST RIVERFRONT DYNAMIC ASIA PACIFIC ETF}: 2016:-1%, 2017:16%, 2018:-8%
{UDR INC., COVANTA HOLDING CORPORATION}: 2016:-1%, 2017:7%, 2018:4%
{NUVEEN MARYLAND QUALITY MUNICIPAL INCOME FUND, NUVEEN HIGH INCOME NOVEMBER 2021 TARGET TERM FUND}: 2016:-2%, 2017:0%, 2018:-2%
{BLACKROCK MUNI INTERMEDIATE DURATION FUND INC, NUVEEN MICHIGAN QUALITY MUNICIPAL INCOME FUND}: 2016:-2%, 2017:0%, 2018:-4%
{VIRTUS LIFESCI BIOTECH CLINICAL TRIALS ETF, SYPRIS SOLUTIONS INC.}: 2016:-33%, 2017:53%, 2018:14%
{NUVEEN QUALITY MUNICIPAL INCOME FUND, PIMCO NEW YORK MUNICIPAL INCOME FUND II}: 2016:-3%, 2017:0%, 2018:-5%
{MFS MUNICIPAL INCOME TRUST, FIRST TRUST ALTERNATIVE ABSOLUTE RETURN STRATEGY ETF}: 2016:-3%, 2017:2%, 2018:-1%
{MAINSTAY MACKAY DEFINEDTERM MUNICIPAL OPPORTUNITIE, PIMCO MUNICIPAL INCOME FUND II}: 2016:-3%, 2017:7%, 2018:0%

Hive:

BLACKROCK MUNIYIELD FUND, INC. -- INVESCO MUNICIPAL TRUST 2016: -5.0, 2017: 2.0, 2018: -5.0
NUVEEN PENNSYLVANIA QUALITY MUNICIPAL INCOME FUND -- NUVEEN SELECT MATURITIES MUNICIPAL FUND 2016: -5.0, 2017: 1.0, 2018: -3.0
CHINA MOBILE (HONG KONG) LTD. -- MFS GOVERNMENT MARKETS INCOME TRUST 2016: -5.0, 2017: -4.0, 2018: -8.0
FIRST TRUST ALTERNATIVE ABSOLUTE RETURN STRATEGY ETF -- MFS MUNICIPAL INCOME TRUST 2016: -4.0, 2017: 3.0, 2018: -2.0
TRIPLEPOINT VENTURE GROWTH BDC CORP. -- PIMCO STRATEGIC INCOME FUND, INC. 2016: -3.0, 2017: 4.0, 2018: 4.0

NUVEEN MICHIGAN QUALITY MUNICIPAL INCOME FUND -- PIMCO NEW YORK MUNICIPAL INCOME FUND II 2016: -3.0, 2017: -1.0, 2018: -5.0
LINCOLN EDUCATIONAL SERVICES CORPORATION -- STURM, RUGER & COMPANY, INC. 2016: -14.0, 2017: 4.0, 2018: 8.0
LEGG MASON DEVELOPED EX-US DIVERSIFIED CORE ETF -- NUVEEN TAX-ADVANTAGED TOTAL RETURN STRATEGY FUND 2016: -1.0, 2017: 22.0, 2018: -3.0
VANGUARD INTERMEDIATE-TERM TREASURY ETF -- ISHARES MBS ETF -- ISHARES GNMA BOND ETF -- VANGUARD MORTGAGE-BACKED SECURITIES ETF -- EATON VANCE HIGH INCOME 2021 TARGET TERM TRUST 2016: -1.0, 2017: 0.0, 2018: -2.0
THE GDL FUND -- BLACKROCK INCOME TRUST INC. (THE) 2016: -1.0, 2017: -2.0, 2018: -7.0

Spark:

((2016: 0%,2017: +1%,2018: +3%),CONSTELLATION ALPHA CAPITAL CORP.; I-AM CAPITAL ACQUISITION COMPANY; "DRAPER OAKWOOD TECHNOLOGY ACQUISITION, INC.")
((2016: 0%,2017: 0%,2018: -72%),SSLJ.COM LIMITED; MENLO THERAPEUTICS INC.)
((2016: 0%,2017: 0%,2018: +9%),ALLIANZGI DIVERSIFIED INCOME & CONVERTIBLE FUND; "COLUMBIA FINANCIAL, INC."; PACER MILITARY TIMES BEST EMPLOYERS ETF)
((2016: 0%,2017: 0%,2018: -25%),"FIVE POINT HOLDINGS, LLC"; "FRONTEO, INC."; MITSUBISHI UFJ FINANCIAL GROUP INC)
((2016: +19%,2017: +9%,2018: +7%),ERIE INDEMNITY COMPANY; POWERSHARES S&P SMALLCAP UTILITIES PORTFOLIO)
((2016: 0%,2017: 0%,2018: -19%),BRIGHTSPHERE INVESTMENT GROUP PLC; ONESMART INTERNATIONAL EDUCATION GROUP LIMITED)
((2016: +16%,2017: -2%,2018: +4%),GUGGENHEIM CREDIT ALLOCATION FUND; "ESSA BANCORP, INC.")
((2016: 0%,2017: -1%,2018: +3%),"GORES HOLDINGS II, INC."; TPG PACE HOLDINGS CORP.; BISON CAPITAL ACQUISITION CORP.)
((2016: +3%,2017: -8%,2018: -15%),"COMPASS MINERALS INTERNATIONAL, INC."; SIGNATURE BANK)

((2016: 0%,2017: 0%,2018: +34%),UNUM THERAPEUTICS INC.; BAYCOM CORP)
((2016: 0%,2017: 0%,2018: +15%),"TARENA INTERNATIONAL, INC."; ATLANTICUS HOLDINGS CORPORATION)

COMMENTO ALLE SCELTE PROGETTUALI E AI RISULTATI

Il linguaggio scelto per i job in MapReduce è stato Java.
Il linguaggio usato per i job 1 e 3 in Spark è stato Scala, mentre per il job 2 in Spark è stato usato Java.
Questo nell'ottica di vedere come cambiava il tempo di esecuzione dei job con un linguaggio ottimizzato per il framework.
Effettivamente come si vede nelle tabelle, la scelta del linguaggio Scala influisce positivamente sui tempi di esecuzione.

Come già anticipato, abbiamo deciso di affrontare i problemi in maniera differente anche per sperimentare e provare soluzioni differenti, e vedere come sarebbe variato il tutto.

Ci siamo dunque divisi il lavoro nel seguente modo:
Costa Danilo ha scritto il Job 1 e 3 in Spark, e il 2 in MapReduce.
Frillici Filippo ha scritto il Job 1 e 3 in MapReduce, e il 2 in Spark.

Per la parte Hive, è stata svolta insieme.

Ognuno si è occupato anche della parte relativa a scrivere lo pseudocodice, e testare l' esecuzione per la stampa dell'output, per i propri job.

Alcune scelte progettuali sono state fatte dopo aver effettuato delle prove.
Ad esempio in MapReduce è stato testato sia l'utilizzo di un Comparatore custom, che di una Collection ordinata.
Venendo più veloce con la Collection, è stato usato quel metodo.

Per quanto riguarda Spark, l’ utilizzo di Collection invece rallentava un pochino, quindi si è scelto di utilizzare più map e reduce in cascata.

Inoltre ove sono state usate risorse esterne, è stata citata la fonte nel codice.

Nel vedere i risultati, ad una prima occhiata si può pensare ad un andamento lineare, poiché raddoppiando l’ input, i tempi aumentano di circa il doppio.
Già andando a quadruplicare l’ input, (sia historical_stock_prices che historical_stocks), la linearità percepita prima va a diminuire, ed è ragionevole pensare che all'aumentare del dataset, cresca con andamento esponenziale.