Sommario

[Introduzione 1](#_Toc439930691)

[Obiettivo del documento 1](#_Toc439930692)

[Finalità del progetto 1](#_Toc439930693)

[Condizioni di analisi 2](#_Toc439930694)

[Considerazioni sull’intervallo temporale 2](#_Toc439930695)

[Cosa sono le tracce di Google 2](#_Toc439930696)

[Analisi 2](#_Toc439930697)

[Task event table 3](#_Toc439930698)

[Task per job 4](#_Toc439930699)

[Struttura del file (.txt) 4](#_Toc439930700)

[Analisi 4](#_Toc439930701)

[Task per machine 12](#_Toc439930702)

[Struttura del file (“start time[h] – machineId – event: event\_type – granularity” .txt - .csv) 12](#_Toc439930703)

[Analisi 13](#_Toc439930704)

[Conclusioni 17](#_Toc439930705)

[Tecnologie utilizzate 17](#_Toc439930706)

[Documentazione 17](#_Toc439930707)

# Introduzione

## Obiettivo del documento

Il presente documento ha come obiettivo la descrizione dell’analisi effettuata sulle tracce di Google, in particolare per quanto riguarda la valutazione dei task per job e dei task per macchina, e la discussione dei risultati ottenuti da tale analisi.

La descrizione sarà prevalentemente testuale con l’integrazione di rappresentazioni grafiche e di porzioni di codice.

## Finalità del progetto

Tale documento è stato redatto per la totalità delle persone che saranno coinvolte nel processo di analisi e integrazione delle tracce.

I dati ottenuti tramite la nostra analisi, concentrata sulla valutazione dei task/job e dei task/machina, verranno poi integrati con dati provenienti da altre elaborazioni delle tracce dello stesso intervallo temporale e con tracce non incluse nel subset temporale in valutazione.

## Condizioni di analisi

Affinché i dati possano essere integrati si sono stabilite convenzioni di analisi:

* Temporale:

la traccia che valuteremo avrà durata di un giorno, 2 Maggio 2011. Si definisce quindi un sottogruppo di tracce grazie ai timestamp

start time = 18600000000 (May 2nd, 2011 – 00:00 EDT)

stop time = 105000000000 (May 2nd, 2011 – 23:59 EDT)

* Input:

<trace>, <condition>

Condition, indica i vincoli imposti nell’analisi

* Output:

numero di task per job e numero di task per macchina. I risultati devono essere stampati in un file di testo.

* Analisi:

statistiche sul numero di task che possono essere runnati contemporanemente nello stesso job e nella stessa macchina.

Sono utilizzate diverse finestre all’interno del subset temporale per identificare trend particolari.

## Considerazioni sull’intervallo temporale

La traccia analizzata ha durata di un giorno, 2 Maggio 2011. Si definisce quindi un sottogruppo di tracce grazie ai timestamp

start time = 18600000000 (May 2nd, 2011 – 00:00 EDT)

stop time = 105000000000 (May 2nd, 2011 – 23:59 EDT)

## Cosa sono le tracce di Google

Le tracce sono considerevoli datasets basati sui cluster di Google, un insieme di macchine connesse in una rete high-bandwidth.

L’ insieme di macchine che condivide il cluster manager viene chiamato *cell*. A ogni macchina sono sottomessi dei *job*, ognuno dei quali è composto da più *task.* Ogni task rappresenta un programma Linux, possibilmente con più processi, runnato su una singola macchina.

A ogni task, job e machina viene associato un ID univoco:

* Il taskID è generato utilizzando in jobID unito a un indice interno. L’ identificativo può rimanere lo stesso quando il task viene stoppato e rischedulato.
* Il jobID generalmente non rimane lo stesso quando il job viene stoppato e rischedulato.
* Il machineID può rimanere lo stesso quando la macchina viene rimossa dal cluster e poi reinserita.

# Analisi

Come specificato nel paragrafo precedente, l’obiettivo della nostra analisi è analizzare statistiche ricorrenti nell’ambito task per job e task per macchina.

Per far ciò abbiamo analizzato la *task event table.*

### Task event table

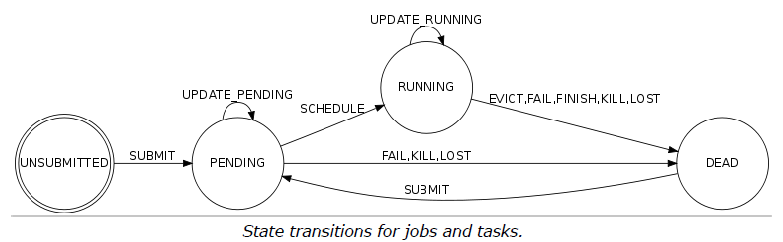
La tabella *task event* contiene i seguenti campi:

1. timestamp
2. missing info
3. *job ID*
4. *task index* - within the job
5. machine ID
6. event type
7. user name
8. scheduling class
9. priority
10. resource request for CPU cores
11. resource request for RAM
12. resource request for local disk space
13. different-machine constraint

I campi che teniamo in considerazioni per l’analisi sono:

* 1. Timestamp, necessario per individuare il subset giornaliero e per individuare le window di analisi. [microsecondi]
* 3. JobID, necessario per individuare il task all’interno del job
* 6. Event type, valori numerici che definiscono lo stato di un task (o di un job):
* SUBMIT (0): Un task (o un job)
* SCHEDULE (1): Un task (o un job) è stato schedulato su una macchina. Il fatto che un task (o un job) sia stato schedulato non significa che stia già runnando. (Per i job si definisce un evento schedule quando un suo task viene schedulato su una macchina.
* EVICT(2): Un task (o un job) viene deschedulato perchè è presente un task (o un job) con priorità più alta.
* FAIL(3): Un task (o un job) viene deschedulato a causa di un failure del task.
* FINISH(4): Un task (o un job) ha completato il suo ciclo normalmente.
* KILL(5): Un task (o un job) viene cancellato perché un task ( o un job) da cui era dipendente viene ucciso.

Di seguito si rappresenta il diagramma di stato



## Task per job

### Struttura del file (.txt)

*startTime[micros],endTime[micros],mean,variance*

* Start time - inizio periodo di valutazione
* End time - fine periodo di valutazione
* Mean - media del numero di task per job nell’ intervallo di tempo [end time – start time]
* Variance - varianza di task per job nell’ intervallo di tempo [end time – start time]

#### Example

|  |  |  |  |
| --- | --- | --- | --- |
| *startTime[micros]* | *endTime[micros]* | *mean* | *variance* |
| *18600000000* | *22200000000* | *21.060357* | *15426.772763* |

### Analisi

Inanzitutto abbiamo parsato i file .csv, immagazzinandoli in un RDD secondo la struttura *[timestamp [micros],jobID,event\_type]* selezionando dalla tabella *task event* solo i campi di nostro interesse, filtrando le tracce secondo l’intervallo di tempo indicato nei constraint.

googleRDD=(sc.textFile(path)

.map(lambda x:x.split(','))

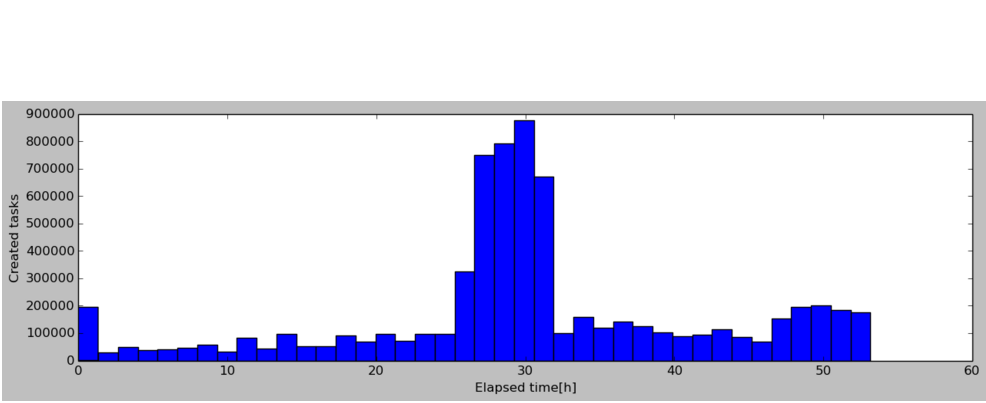
.map(lambda x:(int(x[0]),int(x[2]),int(x[5])))

.filter(lambda x:x[0]>=startTimeMs and x[0]<=endTimeMs))

Su questo insieme di dati abbiamo poi effettuato le analisi.

##### Numero di task generati - Intervallo di 52 ore

Nonostante la consegna indicasse l’analisi temporale di 24 ore, abbiamo preferito riportare anche il grafico del numero di task per job in 52 ore ( circa due giorni) perché sottolinea un aspetto importante dell’analisi, sorto nella fase di generazione del grafico delle 24 ore.



Come si può notare, si individua un intervallo di tempo [25-32 h] in cui si ha un aumento considerevole di task creati (circa 0.9 M).

Confrontiamo questo dato con la media dei task generati dal job nello stesso intervallo di tempo, possiamo notare che il la media è di circa 600 task. Questa discrepanza di valori [0.9 M task – 600 task] ci permette di capire che l’aumento di task generati per job non implica un aumento della media, in quanto è possibile che ci siano un maggiore numero di job che generano in media un minor numero di task.

L’istogramma rende quindi visibile i valori elevati di varianza, che stanno a indicare l’elevata variabilità del fenomeno di generazione dei task.

Per tutti gli istogrammi riportati abbiamo preferito partizionare l’intervallo di tempo in ore, per avere una visione chiara ma al contempo significativa del fenomeno.

#### Numero di task generati per ogni job

Un dato fondamentale per l’analisi è determinare il numero di task per job, dando la possibilità all’utilizzatore di poter scegliere quale tipo di *event type* vuole analizzare.

googleRDD=(sc.textFile(path)

.map(lambda x:x.split(','))

.map(lambda x:(int(x[0]),int(x[2]),int(x[5])))

.filter(lambda x:x[0]>=startTimeMs and x[0]<=endTimeMs)

.filter(lambda x:x[2]==event\_type))

#### Probabilità di tipi di eventi

Per poter analizzare le probabilità su tipi di eventi è stato necessario filtrare l’RDD descritto in precedenza secondo il campo *event\_type.*

Identificando la probabilità come .

* Probabilità che un task venga schedulato (eventtype=3)
* Probabilità che un task vada nello stato evicted (eventtype=2)
* Probabilità che un task venga ucciso (eventtype=5)
* Probabilità che un task riesca a finire il ciclo (eventtype=4)

#### Media e varianza

Definiamo un nuovo RDD che rappresenta un subset del googleRDD precedente, identificando solo i submitted task

dataRDD=googleRDD.filter(lambda x:x[2]==0)

perché l’obiettivo è andare a trovare quale è la media e la varianza di task che sono effettivamente pronti a essere schedulati.

Nei grafici seguenti vengono presentati tramite una spezzata la media e la varianza del numero di task generati nell’intervallo di tempo, abbiamo deciso di dividere questo intervallo di tempo in ore per non perdere significatività e rendere leggibile il grafico.

##### Media

Per trovare la media di task creati per job applichiamo la seguente formula

def meanTaskPerJobDict(jobDict):

if(len(jobDict)==0):

return 0

values=jobDict.values()

s=sum(values)

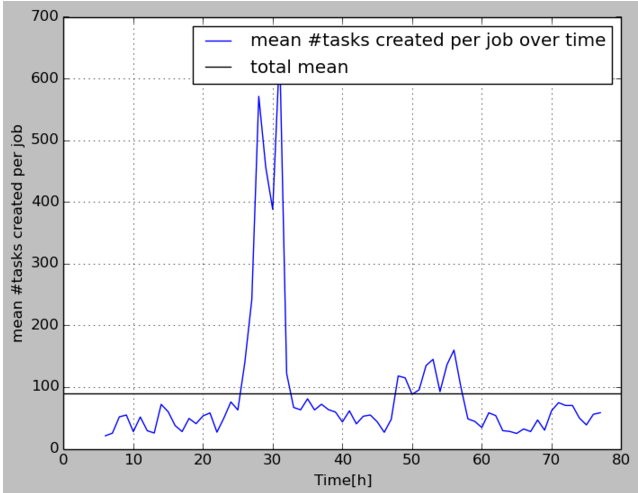
return float(s/len(jobDict))

Il grafico riportato in seguito mostra la relazione che intercorre tra numero di task creati per job nell’intervallo di tempo e la media totale.

I grafici sono espressi in microsecondi.

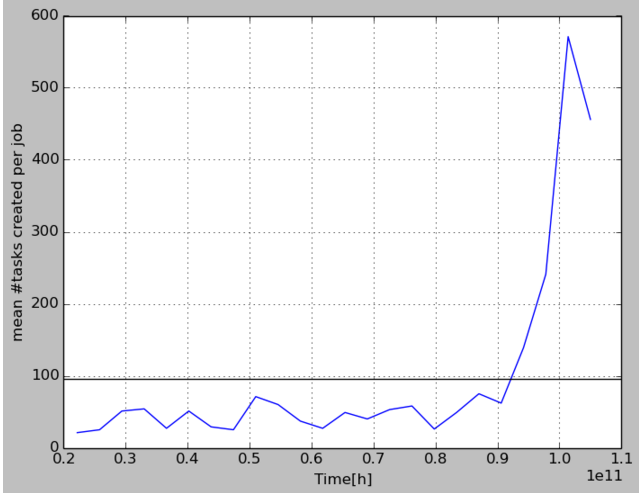
##### Intervallo di 72 ore – granularità un punto ogni ora

Così come per l’istogramma, abbiamo ritenuto opportuno analizzare la variazione della media in un intervallo temporale maggiore di 24 ore, per meglio evidenziare la relazione che intercorre con il numero di task creati.

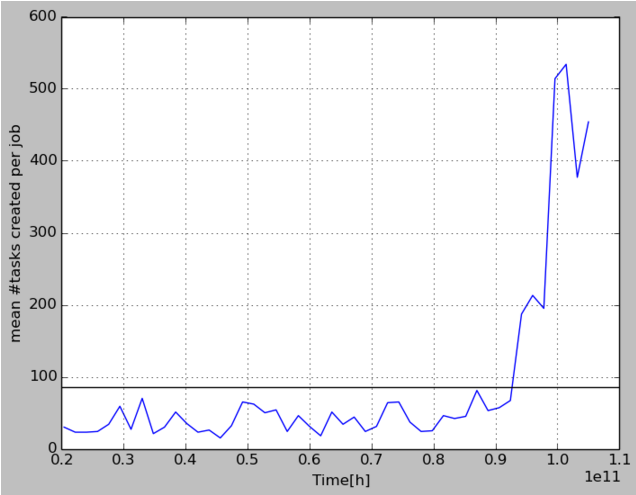


Come si può evincere dal grafico, la media del numero di task creati nelle window di un ora si mantiene vicina alla media totale dei task. Si ha un picco in corrispondenza delle 25 - 32 h come per l’istogramma.

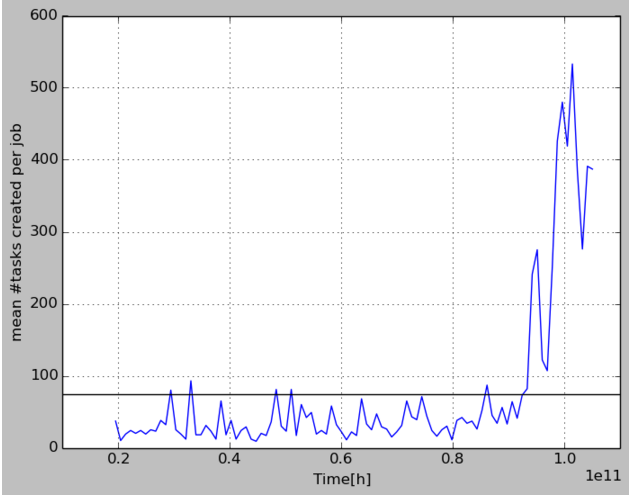
##### Intervallo di 24 ore – granularità un ora



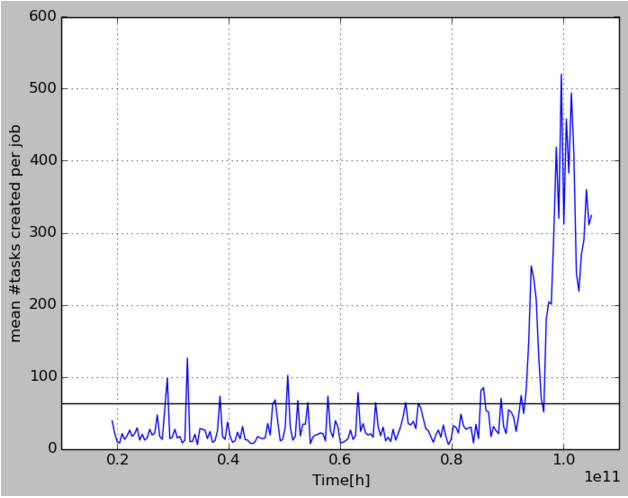
##### Intervallo di 24 ore – granularità mezz’ ora



##### Intervallo di 24 ore – granularità quattro punti in un ora



##### Intervallo di 24 ore – granularità otto punti in un ora



L’identificazione di una granularità più fine dimostra un andamento altalenante della media, con picchi e discese alternate, a significare che a seguito di un assegnazione di task a un job, prima di poter acquisirne altri deve completare quelli assegnati in precedenza.

##### Deviazione standard

La varianza viene definite come

def varianceTaskPerJobDict(jobDict,mean):

if(len(jobDict)==0):

return 0

squareSum=0

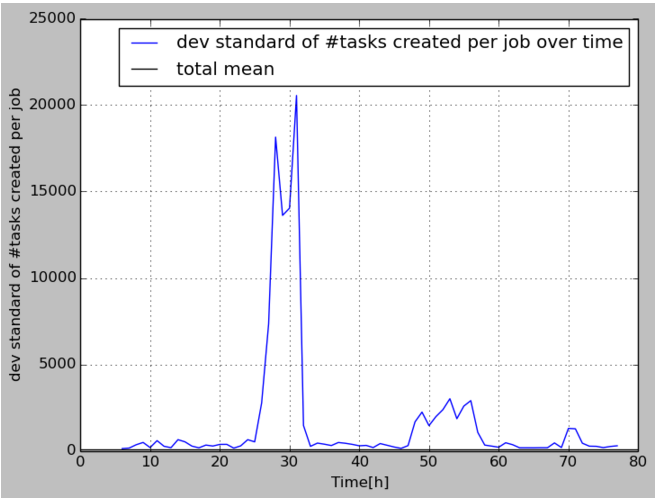
for v in jobDict.values():

squareSum+=pow(v-mean,2)

return float(squareSum/len(jobDict))

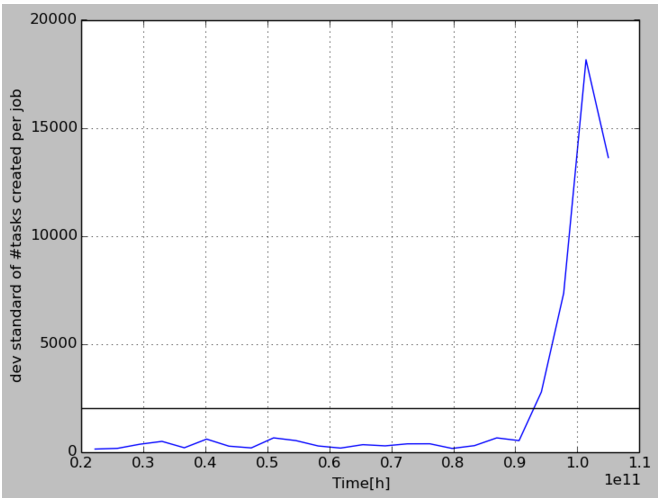
##### Intervallo di 72 ore

Come nel caso della media, abbiamo ritenuto opportuno analizzare la deviazione standard in un intervallo temporale maggiore di 24 ore, per meglio evidenziare la relazione che intercorre con il numero di task creati.

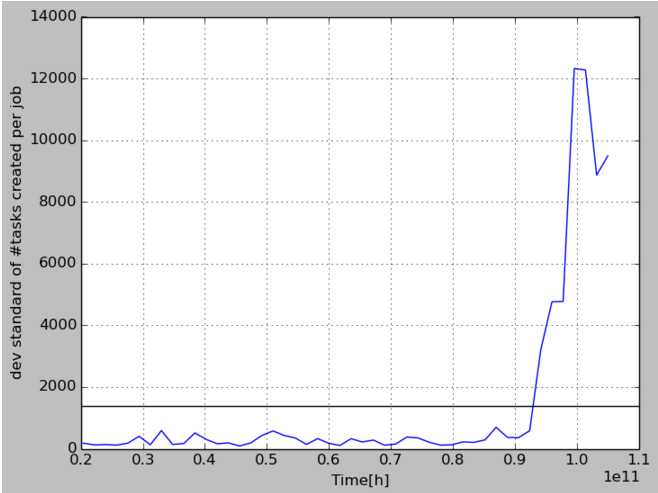


Come si può evincere dal grafico, la deviazione standard del numero di task creati nel tempo si mantiene vicina alla media totale dei task. Si ha un picco in corrispondenza delle 25 - 32 h come per l’istogramma, che evidenzia il comportamento anomalo rispetto alle altre ore in considerazione.

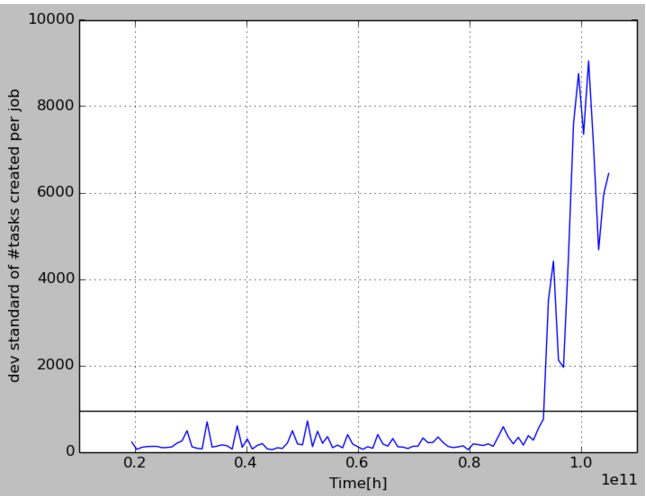
##### Intervallo di 24 ore – granularità un ora



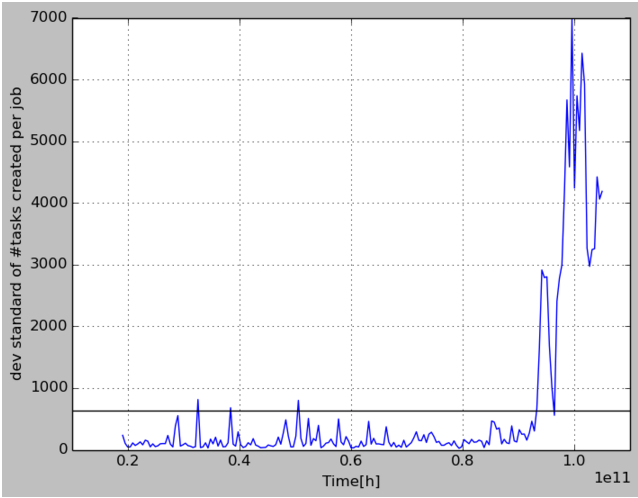
##### Intervallo di 24 ore – granularità mezz’ ora



##### Intervallo di 24 ore – granularità quattro punti in un ora



##### Intervallo di 24 ore – granularità otto punti in un ora



Avendo diminuito l’intervallo temporale complessivo, si è modificato anche il valore di deviazione standard.

Perché LA DEVIAZIONE STADARD CAMBIA?

La discrepanza tra deviazione standard complessivo e deviazione standard con granularità evidenzia come il picco dalle 25 h in poi vada ad aumentare considerevolmente il suo valore totale.

##### Media e varianza nelle window size

Le funzioni media e varianza vengono valutate nei subset temporali di un ora

result=computeData('file:///home/vagrant/data/googleTraces/\*.csv.gz',18600,105000,3600)

L’intervallo temporale e la granularità possono essere modificate.

## Task per machine

### Struttura del file (“start time[h] – machineId – event: event\_type – granularity” .txt - .csv)

*startTime[micros],endTime[micros],mean*

* Start time - inizio periodo di valutazione
* End time - fine periodo di valutazione
* Mean - numero di task (schedulati - completati) per macchina nell’ intervallo di tempo [end time – start time]
* MachineId – identificativo della macchina in considerazione

#### Example - 5L-906113event-4-3600L

Machine Id: 906113

Event type: 4 (finish)

Granularity: 3600

|  |  |  |
| --- | --- | --- |
| *startTime[micros]* | *endTime[micros]* | *mean* |
| 29400000000 | 33000000000 | 2.000000 |

### Analisi

Innanzitutto abbiamo parsato le trace di nostro interesse, ottenendo i machineID e mappando la tupla come *(timestamp [micros],machineID,event\_type).* Calcolato poi il numero di macchine presenti, più di 12.000.

googleRDD=(sc.textFile(path)

.map(lambda x:x.split(','))

.filter(lambda x:x[4]!='')

.map(lambda x:(int(x[0]),int(x[2]),int(x[4]),int(x[5])))

.filter(lambda x:x[1]==machine\_id)

.filter(lambda x:x[0]>=startTimeMs and x[0]<=endTimeMs))

dataRDD=googleRDD.filter(lambda x:x[2]==event\_type)

Inizialmente l’analisi era stata progettata per analizzare per ogni macchina il numero di task creati. Dal momento che avremmo dovuto operare calcoli su enormi quantità di task per la totalità delle macchine, rendendo il costo temporale estremamente grande, abbiamo deciso di focalizzare l’attenzione solo sulle macchine più significative. Per significatività intendiamo le prime 10 macchine che schedulano il maggior numero di task (event\_type = 1).

def findMostUsedMachines(path):

machineRDD=(sc.textFile(path)

.map(lambda x:x.split(','))

.filter(lambda x:x[4]!='')

.map(lambda x:(int(x[4]),int(x[5])))

.filter(lambda x:x[1]==1)

.map(lambda x:(x[0],1))

.reduceByKey(lambda x,y:x+y))

print "First 10 most used machines"

print machineRDD.takeOrdered(10,lambda k:-k[1])

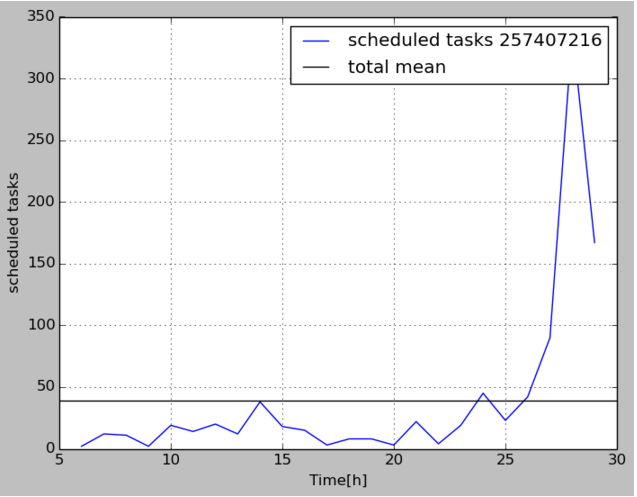
A questo punto abbiamo calcolato il numero di task completati (event\_type = 4) nell’intervallo di tempo per le macchine selezionate, valore che permetterà di trovare il throughput.

Per avere dati di paragone abbiamo calcolato la media del numero di task generati per macchina nell’intervallo di valutazione. Il valore medio per macchina è irrisorio rispetto al numero di task complessivi che sono generati, questo perché siamo in presenza di un grande numero di macchine e quindi a ogni macchina verranno sottomessi in media un basso numero di task, (così come avveniva per i task per job).

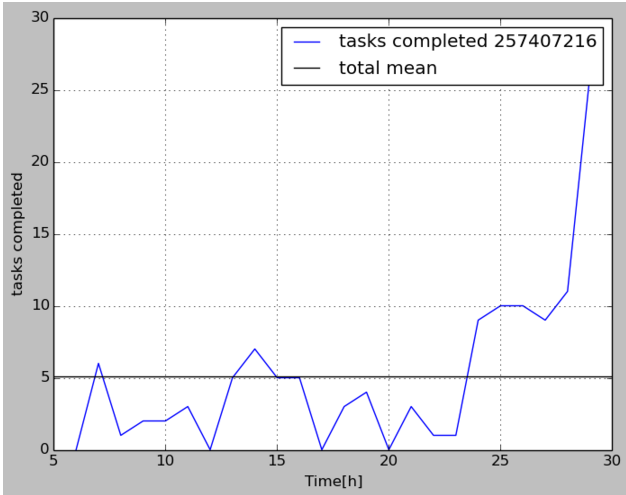
Le due analisi sono poi state unificate graficamente. In seguito sono riportati i grafici di alcune macchine.

##### Macchina 257407216

###### Grafico del numero di task schedulati dalla macchina 257407216

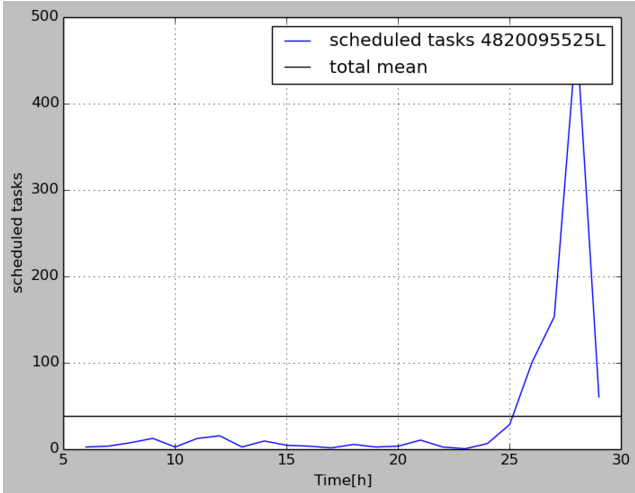


###### Grafico del numero di task completati dalla macchina 257407216

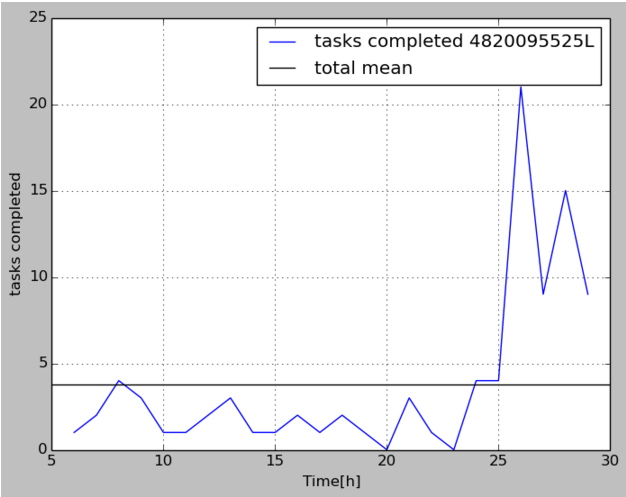


##### Macchina 4820095525

###### Grafico del numero di task schedulati dalla macchina 4820095525

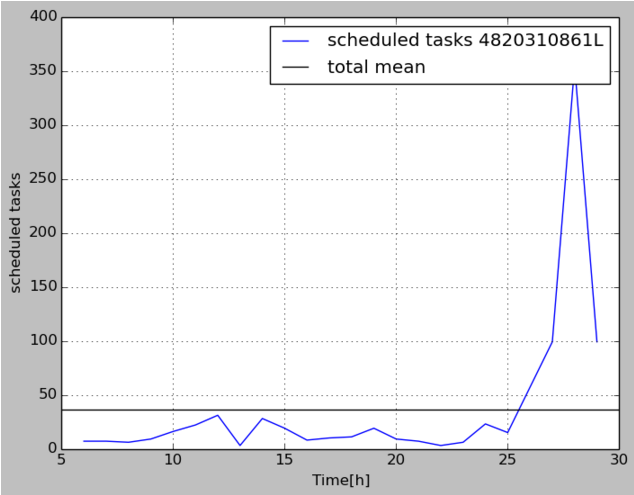


###### Grafico del numero di task completati dalla macchina 4820095525

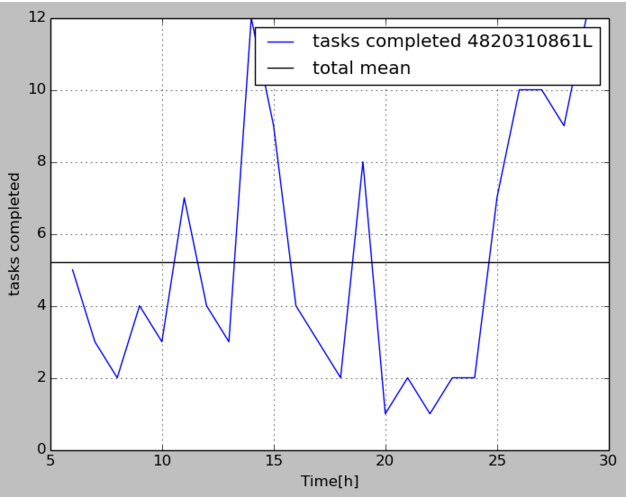


##### Macchina 4820310861

###### Grafico del numero di task schedulati dalla macchina 4820310861



###### Grafico del numero di task completati dalla macchina 4820310861



NOTIAMO CHEIL NUMERO DI TASK COMPLETATI – può essere che dei task vengano completati dopo - è MOLTO BASSO ( 5 RISPERTTO AI 30)????

L’andamento del numero di task assegnati ( e completati) ha un andamento altalenante rispetto alla media altri invece stanno sopra -> non possiamo analizzare un trend!

Evidenziamo che anche i task per macchina seguono l’andamento dell’ istogramma riportato inizialmente, evidenziando un picco per periodi superiori a 25 h.

# Conclusioni

DA FARE

# Tecnologie utilizzate

* Spark
* Phyton

# Documentazione

* ***“Google cluster-usage traces: format + schema”.*** *Charles Reiss, John Wilkes, Joseph Hellerstein. Version of 2013-05-06, for trace version 2. Revised 2014-11-17 for trace version 2.1.*
* ***“Heterogeneity and dynamicity of clouds at scale: Google trace analysis”.*** *Proceedings of the Third ACM Symposium on Cloud Computing. ACM, 2012.*
* *https://commondatastorage.googleapis.com/clusterdata-2011-2/SHA256SUM*

SISTEMARE GRAFICI DELLA DEVIAZIONE STANDARD (DA h PASSA A ms)

SISTEMARE GRAFICI DEL MEAN TIME (DA h PASSA A ms)