Performance Modeling Of Computer Systems And Networks

Relazione Progetto 2017/18

Simone Falvo
smvfal@gmail.com

UNIVERSITA' degli STUDI di ROMA
TOR VERGATA

Indice

1 Introduzione

In questa relazione viene descritta una possibile soluzione ad un problema di edge computing in cui è richiesto di ottimizzare le prestazioni di un sistema di computazione cloud, calibrando i parametri di un algoritmo per l'inoltro di task utente verso un cloudlet ed un server remoto.

Il sistema è in grado di eseguire task di due diverse classi in un cloudlet, situato ad un "hop" di distanza dall'utente, fintanto che le risorse lo consentono, altrimenti vengono inoltrati ed eseguiti in un server remoto. Inoltre, se la somma del numero di job presenti nel cloudlet è uguale al valore di soglia S e se vi è almeno un job di classe 2 in esecuzione, questo viene fatto migrare dal cloudlet al server remoto per far posto ad un task di classe 1 in arrivo, che ha una maggiore "priorità" di esecuzione nel cloudlet.
L'obbiettivo principale è quello di trovare il valore ottimale del parametro S affinché sia minimizzato il tempo di risposta medio.

Il problema è stato affrontato definendo un modello a code per il sistema in modo tale da poter valutare, tramite l'analisi e la simulazione, i tempi di risposta e le altre metriche di performance come il throughput e la popolazione media.
Il modello è stato validato analiticamente tramite lo studio dello stato del sistema a regime, calcolandone la distribuzione stazionaria e valutando le metriche di interesse in relazione ai possibili scenari.
La simulazione è stata realizzata implementando in linguaggio C un programma basato su eventi ed i risultati sono stati raccolti ed elaborati utilizzato il metodo "batch means" per avere una stima del comportamento del sistema a regime.

Nel seguito di questo documento verranno presentati i vari modelli di astrazione del sistema con i risultati che ne derivano, ed infine verrà fatto un cofronto in relazione ai possibili scenari analizzati.

2 Modello Concettuale

Nel modello concettuale il sistema viene descritto come un sistema di code connesse tra loro (figura 1), di quest'ultimo si distinguono i seguenti componenti:

Cloudlet : nodo del sistema che comprende un numero N di server senza coda che operano in parallelo con tassi di servizio specifici per classi di job.

Cloud : nodo del sistema che comprende un numero infinito di server senza coda che operano in parallelo con tassi di servizio specifici per classi di job.

Centro di setup : nodo del sistema che modella la fase di setup di un job interrotto, composto da un numero infinito di server senza coda che operano in parallelo con un tasso di servizio pari a $1/E[S_{setup}]$. Un job interrotto transita per questo centro prima di prendere servizio nel cloud ed il tempo ivi trascorso corrisponde al tempo necessario alla ri-esecuzione del job.

Controllore : componente che implementa la logica di routing in base all'occupazione del cloudlet tenendo conto del parametro di soglia S, non costituisce un centro di servizio in quanto la sua funzione è limitata all'inoltro dei job.

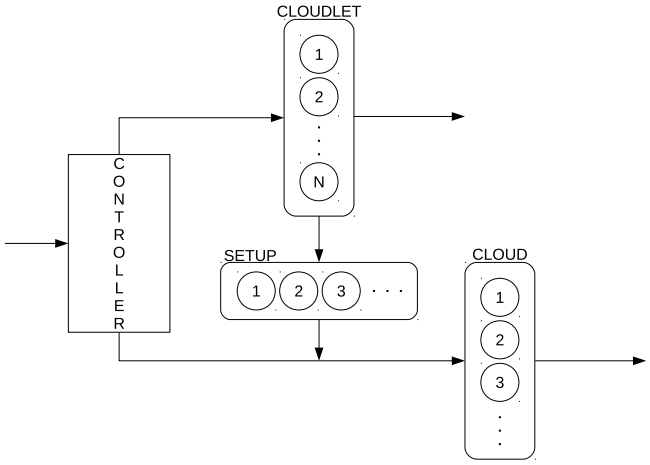

Figura 1: Modello concettuale del sistema

Ogni job che arriva al sistema passa prima per il controllore, il quale decide quale deve essere il nodo di esecuzione (cloud o cloudlet). Se arriva un job di classe 1 ed il cloudlet si trova nella condizione per cui almeno un suo server ha in esecuzione un job di classe 2 e la somma dei job presenti ha raggiunto il valore di soglia S, allora tale server deve sospendere il job e sostituirlo con quello appena arrivato di classe 1. Il job di classe 2 interrotto verrà poi inoltrato al server remoto (cloud), non prima però di aver trascorso un tempo necessario alla propria riesecuzione sul nuovo nodo, tale fase è modellata assumendo che il job venga eseguito in un centro intermedio (centro di setup).

3 Modello di Specifica

In questo modello vengono definite le variabili e le equazioni necessarie al calcolo delle metriche, inoltre viene descritta la logica con cui il sistema evolve a seguito degli eventi che si verificano ed i vincoli a cui esso deve sottostare.

3.1 Metriche

Di principale interesse sono le variabili che compongono lo stato del sistema, ovvero quelle che lo caratterizzano completamente, esse tengonono conto del numero di job in servizio suddivisi per classe e per nodo di esecuzione ad ogni istante di tempo e sono indicate nella tabella 1. Le altre variabili utili al calcolo

$n_1^{clet}(t)$	numero di job di classe 1 nel cloudlet al tempo t
$n_2^{clet}(t)$	numero di job di classe 2 nel cloudlet al tempo t
$n_1^{cloud}(t)$	numero di job di classe 1 nel cloud al tempo t
$n_2^{cloud}(t)$	numero di job di classe 2 nel cloud al tempo t
$n_{setup}(t)$	numero di job in fase di setup al tempo t

Tabella 1: Variabili che compongono lo stato del sistema

delle statistiche del sistema sono indicate nella tabella 2 e riguardano i tempi di risposta ed i completamenti, sempre suddivise per classe e nodo di esecuzione.

$s_{1,i}^{clet}$	tempo di servizio dell'i-esimo job di classe 1 eseguito nel cloudlet
$s_{2,i}^{clet}$	tempo di servizio dell'i-esimo job di classe 2 eseguito nel cloudlet
$s_{1,i}^{cloud}$	tempo di servizio dell'i-esimo job di classe 1 eseguito nel cloud
$s_{2}^{cloud,i}$	tempo di servizio dell'i-esimo job di classe 2 eseguito nel cloud
$s_{intr,i}^{clet}$	tempo di servizio nel cloudlet dell'i-esimo job interrotto
$s_{intr,i}^{cloud}$	tempo di servizio nel cloud dell'i-esimo job interrotto
s_{i}^{setup}	tempo di setup dell'i-esimo job interrotto
$c_1^{clet}(t)$	numero di job di classe 1 completati nel cloudlet al tempo t
$c_2^{clet}(t)$	numero di job di classe 2 completati nel cloudlet al tempo t
$c_1^{cloud}(t)$	numero di job di classe 1 completati nel cloud al tempo t
$c_2^{cloud}(t)$	numero di job di classe 2 completati nel cloud al tempo t
$n_{intr}(t)$	numero di job interrotti al tempo t

Tabella 2: Variabili tempi di servizio e completamenti

3.2 Vincoli

Una volta definito lo stato del sistema e le variabili che lo compongono è necessario definire anche come esse sono relazionate ed i vincoli a cui sono sottoposte.

1. Ad ogni istante di tempo t devono valere le seguenti condizioni:

$$\begin{aligned} n_1^{clet}(t) + n_2^{clet}(t) &\leq N \\ n_1^{clet}(t) + n_2^{clet}(t) &\leq S \qquad \text{se } n_2^{clet}(t) > 0 \end{aligned}$$

2. Stato iniziale ($t = t_{start}$):

$$\begin{aligned} n_1^{clet}(t) = n_2^{clet}(t) = n_1^{cloud}(t) = n_2^{cloud}(t) = n_{setup}(t) &= 0 \\ c_1^{clet}(t) = c_2^{clet}(t) = c_1^{cloud}(t) = c_2^{cloud}(t) = c_{setup}(t) &= 0 \end{aligned}$$

3. Il primo evento deve essere un arrivo

4. Dopo l'arrivo di un numero prefissato di job, i successivi arrivi vengono ignorati e non più processati. Il processo di arrivo si interrompe quando viene processato un numero prefissato di job. L'ultimo evento corrisponde al completamento dell'ultimo job.

5. La selezione dei server all'interno di un nodo non è regolata da nessun algoritmo, poiché metriche relative ai singoli server non sono rilevanti ai fini dell'applicazione.

3.2.1 Tempi di risposta

Poiché i nodi del sistema sono sprovvisti di code, i tempi medi di risposta corrispono ai tempi medi di servizio, il calcolo viene quindi ridotto al rapporto tra la somma dei tempi di servizio sperimentati dai job in esame e la loro quantità, pertanto, per prima cosa è utile calcolare le somme dei tempi:

$$s_j^{clet} = \sum_{i=1}^{c_j^{clet}(t_{stop})} s_{j,i}^{clet} \qquad s_j^{cloud} = \sum_{i=1}^{c_j^{cloud}(t_{stop})} s_{j,i}^{cloud} \qquad j = 1,2$$

$$s_{intr} = \sum_{i=1}^{n_{intr}(t_{stop})} (s_{intr,i}^{clet} + s_{intr,i}^{cloud} + s_i^{setup})$$

Con queste formule si è ottenuta la somma dei tempi di servizio dei job suddivisi per classe e nodo di esecuzione, ed anche una relativa esclusivamente ai job interrotti in cui vengono sommati i tempi associati ai differenti nodi che percorrono (cloudlet, setup e cloud).

È importante notare che il numero di job coinvolti è relativo ad un'istante di tempo corrispondente alla fine della finestra di osservazione $[t_{start}; t_{stop}]$, ove si è posto per semplicità $t_{start} = 0$.

Adesso per il calcolo dei tempi di risposta rimane da fare il rapporto con i completamenti relativi ai nodi oppure alle classi di interesse.

$$E[T_j^{clet}] = E[S_j^{clet}] = \frac{s_j^{clet}}{c_j^{clet}(t_{stop})} \qquad j = 1,2 \tag{1}$$

$$E[T_j^{cloud}] = E[S_j^{cloud}] = \frac{s_j^{cloud}}{c_j^{cloud}(t_{stop})} \qquad j = 1,2 \tag{2}$$

$$E[T_{intr}] = E[S_{intr}] = \frac{s_{intr}}{n_{intr}(t_{stop})} \tag{3}$$

$$E[T_1] = E[S_1] = \frac{s_1^{clet} + s_1^{cloud}}{c_1^{clet}(t_{stop}) + c_1^{cloud}(t_{stop})} \tag{4}$$

$$E[T_2] = E[S_2] = \frac{s_1^{clet} + s_1^{cloud} + s_{intr}}{c_2^{clet}(t_{stop}) + c_2^{cloud}(t_{stop})} \tag{5}$$

$$E[T] = E[S] = \frac{s_1^{clet} + s_1^{cloud} + s_2^{clet} + s_2^{cloud} + s_{intr}}{c_1^{clet}(t_{stop}) + c_1^{cloud}(t_{stop}) + c_2^{clet}(t_{stop}) + c_2^{cloud}(t_{stop})} \tag{6}$$

Nella formula 3 la somma dei tempi di servizio dei job interrotti viene divisa per il numero di interruzioni, che equivale al numero di completamenti.
Nella formula 5, in cui si calcola il tempo di risposta dei job di classe 2, vengono considerati i job della suddetta classe che passano esclusivamente per il cloudlet e per il cloud oltre ai job che subiscono le interruzioni, ma al denominatore non è presente il numero di job interrotti n_{intr} perché è già incluso nella variabile c_2^{cloud}, essendo tali job completati nel cloud. Lo stesso discorso vale per la formula 6.

3.2.2 Popolazione media

La popolazione media viene calcolata integrando le variabili in un'intervallo di tempo corrispondente alla finestra di osservazione della simulazione e dividendo per la lunghezza di quest'ultima. Le metriche globali possono essere calcolate sommando le opportune metriche locali, poiché sono relative tutte allo stesso intervallo di tempo.

$$E[N_j^{clet}] = \frac{1}{t_{stop} - t_{start}} \int_{t_{start}}^{t_{stop}} n_j^{clet}(t)\ dt \qquad j = 1, 2 \tag{7}$$

$$E[N_j^{cloud}] = \frac{1}{t_{stop} - t_{start}} \int_{t_{start}}^{t_{stop}} n_j^{cloud}(t)\ dt \qquad j = 1, 2 \tag{8}$$

$$E[N_{setup}] = \frac{1}{t_{stop} - t_{start}} \int_{t_{start}}^{t_{stop}} n_{setup}(t)\ dt \tag{9}$$

$$E[N_1] = E[N_1^{clet}] + E[N_1^{cloud}] \tag{10}$$

$$E[N_2] = E[N_2^{clet}] + E[N_2^{cloud}] + E[N_{setup}] \tag{11}$$

$$E[N_{clet}] = E[N_1^{clet}] + E[N_2^{clet}] \tag{12}$$

$$E[N_{cloud}] = E[N_1^{cloud}] + E[N_2^{cloud}] \tag{13}$$

$$E[N] = E[N_{cloud}] + E[N_{clet}] + E[N_{setup}] \tag{14}$$

$$= E[N_1] + E[N_2] \tag{15}$$

Può essere utile ricordare che i job interrotti sono job di classe 2, pertanto nel calcolo della popolazione media dei job di questa classe nel sistema (equazione 11), vanno considerati anche i job nel centro di setup.

3.2.3 Throughput

Il throughput viene calcolato come il numero di job completati in un intervallo di tempo corrispondente alla finestra di osservazione della simulazione.
Come per la popolazione media è sufficiente comporre i vari throughput locali per ottenere i throughput del sistema, dei singoli nodi oppure delle singole classi. Per esempio per il calcolo del throghput del sistema (equazione 21) si possono sommare i throughput delle classi oppure i throughput dei nodi.

$$X_j^{clet} = \frac{c_j^{clet}(t_{stop})}{t_{stop} - t_{start}} \qquad j = 1, 2 \tag{16}$$

$$X_j^{cloud} = \frac{c_j^{cloud}(t_{stop})}{t_{stop} - t_{start}} \qquad j = 1, 2 \tag{17}$$

$$X_j = X_j^{clet} + X_j^{cloud} \qquad j = 1, 2 \tag{18}$$

$$X_{clet} = X_1^{clet} + X_2^{clet} \tag{19}$$

$$X_{cloud} = X_1^{cloud} + X_2^{cloud} \tag{20}$$

$$X = X_1 + X_2 = X_{clet} + X_{cloud} \tag{21}$$

Il throughput del centro di setup non è particolarmente interessante perché è un nodo interno che non emette job all'esterno del sistema, funge solo da nodo intermedio tra cloudlet e cloud, in definitiva il suo throughput non contriubuisce a quello del sistema.

3.3 Eventi

Lo stato del sistema evolve a seguito di eventi di vario tipo:

1. Arrivo di un job di classe 1
2. Arrivo di un job di classe 2
3. Partenza di un job
4. Setup

L'algoritmo 1 mostra tale evoluzione e le azioni che vengono intraprese in ogni possibile caso.
In generale ad ogni arrivo viene stabilito il nodo di esecuzione in base allo stato del cloudlet e vengono aggiornate opportunamente le variabili, nel caso si debba sostituire un job di classe 2 con uno di classe 1 appena arrivato si provvede a rimuovere il relativo tempo di servizio $s_2^{clet,k}$ precedentemente aggiunto e a registrare l'ammontare di tempo $s_{intr,k}$ per cui il job è stato in esecuzione prima che venisse interrotto. Nel caso in cui si verifica un evento di partenza viene incrementato la corrispondente variabile di completamento e nel caso di un evento di setup, il relativo job precedentemente interrotto può essere finalmente eseguito sul cloud. Si noti anche che, nell'aggiornamento delle variabili temporali, l'istante t' è corrisponde al momento in cui si verifica l'evento successivo a quello corrente che avviene all'istante t.

4 Modello Computazionale

Il modello computazionale consiste in un programma di simulazione di tipo next-event che impiega il metodo "batch means" per il calcolo di tutte le metriche di interesse. A questo livello di astrazione del sistema si passa ad implementare tutto ciò che è stato definito formalmente nel modello di specifica, in particolare in questa sezione verranno descritte le strutture dati impiegate per la rappresentazione delle variabili, le funzioni ed i costrutti che realizzano la logica del sistema, come vengono generati i dati di input alla simulazione ed infine le metodologie con cui vengono collezionati ed elaborati i dati di output.

La simulazione viene eseguita un numero R di volte in modo da ottenere un campione di tale dimensione con cui generare un intervallo di confidenza al 95% per le statistiche ottenute. Le replicazioni della simulazione sono gestite con un ciclo for all'inizio del quale vengono re-inizializzate tutte le variabili e viene reimpostato il seme per il PRNG affinché le distribuzioni generate in ogni replicazione siano indipendenti. In ogni replicazione i valori delle variabili vengono scritti in modo iterativo su dei file di output che vengono poi elaborati tramite un successivo programma per il calcolo delle metriche di interesse.

Algorithm 1 Logica del sistema in base agli eventi

Arrivo di un job i di classe 1:
if $n_1^{clet}(t) = N$ **then**
esecuzione su cloud
$s_1^{cloud} \leftarrow s_1^{cloud} + s_{1,i}^{cloud}$
$n_1^{cloud}(t') \leftarrow n_1^{cloud}(t) + 1$
else if $n_1^{clet}(t) + n_2^{clet}(t) < S$ **then**
esecuzione su cloudlet
$s_1^{clet} \leftarrow s_1^{clet} + s_{1,i}^{clet}$
$n_1^{clet}(t') \leftarrow n_1^{clet}(t) + 1$
else if $n_2^{clet}(t) > 0$ **then**
interruzione e setup job k di classe 2
esecuzione su cloudlet job i di classe 1
$s_1^{clet} \leftarrow s_1^{clet} + s_{1,i}^{clet}$
$s_2^{clet} \leftarrow s_2^{clet} - s_2^{clet,k}$
$s_{intr} \leftarrow s_{intr} + s_{intr,k}$
$s_{setup} \leftarrow s_{setup} + s_{setup,k}$
$n_1^{clet}(t') \leftarrow n_1^{clet}(t) + 1$
$n_2^{clet}(t') \leftarrow n_2^{clet}(t) - 1$
else
esecuzione su cloudlet
$s_1^{clet} \leftarrow s_1^{clet} + s_{1,i}^{clet}$
$n_1^{clet}(t') \leftarrow n_1^{clet}(t) + 1$
end if
Arrivo di un job i di classe 2:
if $n_1^{clet}(t) + n_2^{clet}(t) \geq S$ **then**
esecuzione su cloud
$s_2^{cloud} \leftarrow s_2^{cloud} + s_2^{cloud,i}$
$n_2^{cloud}(t') \leftarrow n_2^{cloud}(t) + 1$
else
esecuzione su cloudlet
$s_2^{clet} \leftarrow s_2^{clet} + s_{2,i}^{clet}$
$n_2^{clet}(t') \leftarrow n_2^{clet}(t) + 1$
end if
Partenza di un job di classe j dal cloudlet:
$c_j^{clet}(t') \leftarrow c_j^{clet}(t) + 1$
Partenza di un job di classe j dal cloud:
$c_j^{cloud}(t') \leftarrow c_j^{cloud}(t) + 1$
Setup:
esecuzione su cloud
$s_2^{cloud} \leftarrow s_2^{cloud} + s_2^{cloud,i}$
$n_2^{cloud}(t') \leftarrow n_2^{cloud}(t) + 1$

4.1 Strutture Dati

In primo luogo, vengono definite le strutture dati riguardanti la lista degli eventi possibili ed il clock che regola il tempo di simulazione, successivamente quelle che contegono i dati di output ed infine la struttura dati relativa alla principale

entità manipolata nel sistema: i task (job).

4.1.1 Eventi e Clock Virtuale

Ad ogni istante la lista di eventi è così composta:

- prossimo arrivo job di classe 1
- prossimo arrivo job di classe 2
- al più N completamenti di job nel cloudlet
- 0 o più completamenti di job nel cloud
- 0 o più completamenti di fase di setup dei job interrotti

non essendovi un numero finito di eventi da gestire, occorre realizzare la lista di eventi tramite una struttura dati dinamica, pertanto gli eventi vengono gestiti tramite una coda prioritaria, ordinata per scadenza, ovvero con una politica del tipo Least Remaining Time (figura 2).

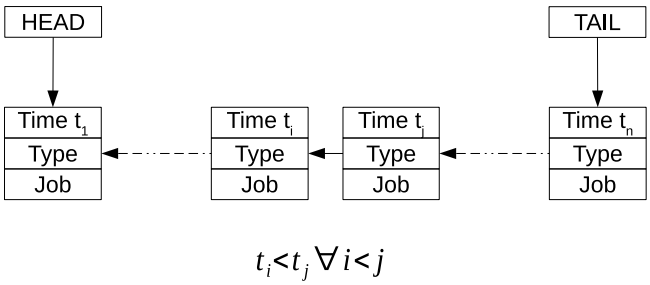

Figura 2: Coda prioritaria di eventi, ordinata per istante di scadenza

Un generico evento è composto dai seguenti campi:

- L'istante in cui l'evento si verifica
- La tipologia di evento (arrivo, partenza, setup)
- Il job associato
- Un array contenente lo stato del sistema (la sua struttura verrà discussa più avanti)

Ogniqualvolta viene creato un evento, questo viene inserito nella coda nella posizione opportuna tramite un'operazione di *enqueue*, affinché ogni operazione di *dequeue* estragga l'evento più imminente.

Il tempo di simulazione è regolato da un clock virtuale che tiene conto dell'istante corrente e del successivo in modo tale da poter considerare intervalli di tempo utili per il calcolo di statistiche di tipo time-averaged.

basic.h

```
struct event {
    double time;
    struct job_t job;
    unsigned int type;
    unsigned int n[4];
};

typedef struct {            /* simulation clock                    */
    double current;         /*    current time                     */
    double next;            /*    next (most imminent) event time  */
} clock;
```

4.1.2 Variabili di Output

I dati che man mano devono essere raccolti durante la simulazione sono memorizzati in variabili suddivise in base alla tipologia e al nodo di esecuzione dei job. Per esempio, lo stato del sistema, così come il numero di arrivi e completamenti, è implementato tramite un array di dimensione 4, di cui ogni slot corrisponde ad una combinazione classe-nodo a cui il job può essere associato.

L'accesso ad uno slot dell'array viene effettuato tramite un indice che viene calcolato tramite la somma di macro che rappresentano le varie combinazioni, infatti, se si vuole considerare il numero di job di classe 1 in esecuzione nel cloud basta accedere all'array tramite l'indice che deriva dalla somma delle macro J_CLASS1 e CLOUD. La tabella X descrive le possibili combinazioni di macro legate all'indice di accesso.

[TABELLA X]

4.1.3 Task

Un generico task, anche detto job, viene implementato come una struttura specifica dotata dei seguenti attributi:

id : identificatore univoco necessario ad effettuare un riordino dei job nel momento in cui vanno considerate le statistiche in un ordine corrispondente agli istanti di arrivo dei singoli job

class : specifica la classe del job, può assumere il valori J_CLASS1 e J_CLASS2 (macro definite nel file di configurazione);

node : specifica il nodo in cui il job è in esecuzione, può assumere i valori CLET, CLOUD, SETUP (macro definite nel file di configurazione), che corrispondono rispettivamente a cloudlet, cloud e nodo di setup;

service : array che memorizza i tempi di risposta seguendo la stessa regola delle combinazioni che riguarda anche lo stato del sistema. Un job di classe 1 in esecuzione nel cloud avrà un tempo di risposta non nullo nello slot relativo, un job interrotto avrà un tempo di risposta non nullo sia nello slot riguardante il nodo cloudlet che in quello riguardante il nodo cloud. setup: tempo che un job trascorre in fase di setup. Il valore rimane nullo nel caso in cui il job non viene interrotto.

basic.h

```
struct job_t {
    unsigned long id;
    unsigned int class;
    unsigned int node;
    double service[4];
    double setup;
};
```

4.2 Generazione dell'input

I dati di input della simulazione, corrispondenti ai tempi di interarrivo e di servizio dei singoli job, vengono generati a runtime in base alle informazioni note sulle relative distribuzioni esponenziali. Per ottenere tali distribuzioni sono state utilizzate le funzioni delle librerie rvgs e rngs di *Steve Park* e *Dave Geyer* descritte in [1].

La funzione *GetArrival()*, ogniqualvolta viene chiamata, restituisce il più imminente istante di arrivo tra un job di classe 1 ed uno di classe 2 e memorizza nella variabile j, passata per riferimento, la classe del job in questione. Tali istanti di arrivo vengono calcolati progressivamente a seguito della generazione dei tempi di interarrivo tra un job e l'altro, più precisamente, non appena viene restituito un istante di arrivo relativo al job di una classe, viene calcolato il successivo per la medesima classe.

La funzione *GetService()* restituisce un valore che deriva dalla generazione di un tempo di servizio esponenziale con media stabilita in base ai parametri j e n passati come argomento che indicano rispettivamente la classe ed il nodo di esecuzione del job.

La funzione *GetSetup()* restituisce semplicemente un valore generato a partire da una distribuzione esponenziale di media $E[S_{setup}]$.

Le funzioni in questione sono elencate di seguito. Si noti che prima di ogni chiamata alle funzioni della libreria *rvgs* viene selezionato, tramite la funzione *SelectStream()*, un flusso di generazione di numeri pseudo-casuali distinto, affinché sia garantita il più possibile l'indipendenza tra le sequenze di numeri generate.

5 Modello Analitico

(figura 3)

6 Conclusioni

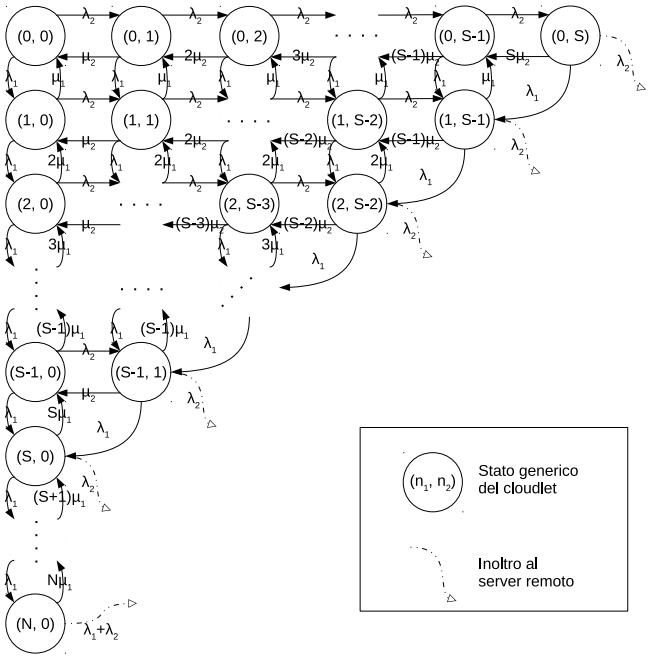

Figura 3: CTMC cloudlet

Riferimenti bibliografici

[1] Leemis L.M., Park S.K. : *"Discrete-Event Simulation: A First Course."* Pearson (2006)