

# Performance Modeling Of Computer Systems And Networks

Relazione Progetto 2017/18

Simone Falvo  
smvfal@gmail.com



## Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
<b>2</b>	<b>Modello Concettuale</b>	<b>3</b>
<b>3</b>	<b>Modello di Specifica</b>	<b>4</b>
3.1	Variabili . . . . .	4
3.2	Vincoli . . . . .	5
3.3	Metriche . . . . .	6
3.3.1	Tempi di risposta . . . . .	6
3.3.2	Popolazione media . . . . .	7
3.3.3	Throughput . . . . .	7
3.3.4	Interruzioni . . . . .	8
3.4	Eventi . . . . .	8
<b>4</b>	<b>Modello Computazionale</b>	<b>9</b>
4.1	Strutture dati . . . . .	9
4.1.1	Eventi e clock virtuale . . . . .	9
4.1.2	Variabili di Output . . . . .	11
4.1.3	Task . . . . .	11
4.2	Generazione dell'input . . . . .	12
4.3	Flusso principale . . . . .	14
4.3.1	Gestione degli eventi . . . . .	15
4.4	Produzione ed Elaborazione dell'Output . . . . .	15
4.4.1	Tempo di Servizio . . . . .	15
4.4.2	Throughput, popolazione, e percentuale di interruzioni . .	17
<b>5</b>	<b>Modello Analitico</b>	<b>17</b>
5.1	Calcolo della distribuzione stazionaria . . . . .	17
5.2	Probabilità preliminari . . . . .	19
5.3	Throughput . . . . .	20
5.4	Tempo di risposta: Cloudlet . . . . .	20
5.5	Tempo di risposta: Cloud . . . . .	21
5.6	Tempo di risposta: Sistema . . . . .	22
5.7	Popolazione media: Cloudlet . . . . .	23
5.8	Popolazione media: Cloud . . . . .	23
5.9	Popolazione media: Setup . . . . .	23
5.10	Popolazione media: Sistema . . . . .	23
<b>6</b>	<b>Risultati</b>	<b>24</b>
6.1	Scenario 1: $S = N = 20$ . . . . .	26
6.1.1	Tempi di risposta . . . . .	26
6.1.2	Popolazione . . . . .	26
6.1.3	Throughput . . . . .	26
6.2	Scenario 2: $S = \frac{N}{2} = 10$ . . . . .	26
6.2.1	Tempi di risposta . . . . .	26
6.2.2	Popolazione . . . . .	26
6.2.3	Throughput . . . . .	26
6.3	Scenario 3: $S = \frac{3}{4}N = 15$ . . . . .	26
6.3.1	Tempi di risposta . . . . .	26

<i>INDICE</i>	2
6.3.2 Popolazione . . . . .	26
6.3.3 Throughput . . . . .	26
6.4 Scenario 4: $\mathbf{S} = \frac{\mathbf{N}}{4} = \mathbf{5}$ . . . . .	26
6.4.1 Tempi di risposta . . . . .	26
6.4.2 Popolazione . . . . .	26
6.4.3 Throughput . . . . .	26
<b>7 Conclusioni</b>	<b>26</b>

## 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'obiettivo 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 utilizzando 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 confronto 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.

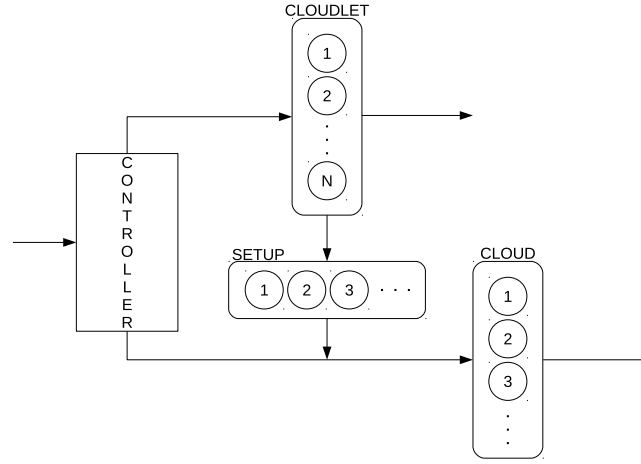


Figura 1: Modello concettuale del sistema

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 Variabili

Di principale interesse sono le variabili che compongono lo stato del sistema, ovvero quelle che lo caratterizzano completamente, esse tengono 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
$s_{2,i}^{cloud}$	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:

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

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.

### 3.3 Metriche

#### 3.3.1 Tempi di risposta

Poiché i nodi del sistema sono sprovvisti di code, i tempi medi di risposta corrispondono 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:

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

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

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.

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

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

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

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

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

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

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.

### 3.3.2 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.

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

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

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

$$E[N_1] = E[N_1^{clet}] + E[N_1^{cloud}] \quad (10)$$

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

$$E[N_{clet}] = E[N_1^{clet}] + E[N_2^{clet}] \quad (12)$$

$$E[N_{cloud}] = E[N_1^{cloud}] + E[N_2^{cloud}] \quad (13)$$

$$E[N] = E[N_{cloud}] + E[N_{clet}] + E[N_{setup}] \quad (14)$$

$$= E[N_1] + E[N_2] \quad (15)$$

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.

### 3.3.3 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.

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

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

$$X_j = X_j^{clet} + X_j^{cloud} \quad j = 1, 2 \quad (18)$$

$$X_{clet} = X_1^{clet} + X_2^{clet} \quad (19)$$

$$X_{cloud} = X_1^{cloud} + X_2^{cloud} \quad (20)$$

$$X = X_1 + X_2 = X_{clet} + X_{cloud} \quad (21)$$

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 contribuisce a quello del sistema.

### 3.3.4 Interruzioni

La percentuale di job interrotti viene calcolata sia rispetto al numero di job di classe 2 del sistema, sia rispetto al numero di job di classe 2 che passano per il cloudlet.

$$P_{intr} = \frac{n_{intr}(t_{stop})}{c_2^{clet}(t_{stop}) + c_2^{cloud}(t_{stop})} \quad (22)$$

$$P_{intr}^{clet} = \frac{n_{intr}(t_{stop})}{n_{intr}(t_{stop}) + c_2^{clet}(t_{stop})} \quad (23)$$

## 3.4 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 ed infine le metodologie con cui vengono collezionati ed elaborati i dati di output.

### 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: il 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).

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)

---

**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* $s_1^{clet} \leftarrow s_1^{clet} + s_{1,i}^{clet}$  $s_2^{clet} \leftarrow s_2^{clet} - s_2^{clet,k}$  $s_{intr} \leftarrow s_{intr} + s_{intr,k}$  $s_{setup} \leftarrow s_{setup} + s_{setup,k}$  $n_{setup}(t') \leftarrow n_{setup}(t) + 1$  $n_1^{clet}(t') \leftarrow n_1^{clet}(t) + 1$  $n_2^{clet}(t') \leftarrow n_2^{clet}(t) - 1$ **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* $s_2^{cloud} \leftarrow s_2^{cloud} + s_{2,i}^{cloud}$  $n_2^{cloud}(t') \leftarrow n_2^{cloud}(t) + 1$ **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:** $c_j^{clet}(t') \leftarrow c_j^{clet}(t) + 1$  $n_j^{clet}(t') \leftarrow n_j^{clet}(t) - 1$ **Partenza di un job di classe  $j$  dal cloud:** $c_j^{cloud}(t') \leftarrow c_j^{cloud}(t) + 1$  $n_j^{cloud}(t') \leftarrow n_j^{cloud}(t) - 1$ **Setup:***esecuzione su cloud* $s_2^{cloud} \leftarrow s_2^{cloud} + s_{2,i}^{cloud}$  $n_{setup}(t') \leftarrow n_{setup}(t) - 1$  $n_2^{cloud}(t') \leftarrow n_2^{cloud}(t) + 1$ 

---

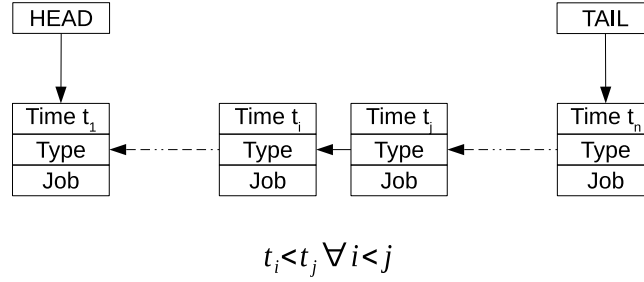


Figura 2: Coda prioritaria di eventi, ordinata per istante di scadenza

Ogniquale volta 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.

#### Implementazione Evento e Clock Virtuale (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` ( $0 + 2 = 2$ ). La tabella 3 descrive le possibili combinazioni di macro legate all'indice di accesso.

#### 4.1.3 Task

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

	CLET	CLOUD	SETUP
J_CLASS1	0	2	
J_CLASS2	1	3	4

Tabella 3: Combinazioni per il calcolo dell'indice degli array

## Definizione macro Nodi e Classi (basic.h)

```

#define J_CLASS1    0          /* job class type 1 */
#define J_CLASS2    1          /* job class type 2 */
#define CLET        0          /* cloudlet index    */
#define CLOUD       2          /* cloud index       */
#define SETUP       3          /* setup index       */

```

**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 rispettive 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()*, ogniquale volta 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.

basic.h

```
double GetArrival(unsigned int *j)
{
    const double mean[2] = {1/L1, 1/L2};
    static double arrival[2] = {START, START};
    static int init = 1;
    double temp;

    if (init) {
        SelectStream(0);
        arrival[0] += Exponential(mean[0]);
        SelectStream(1);
        arrival[1] += Exponential(mean[1]);
        init=0;
    }

    if (arrival[0] <= arrival[1])
        *j = 0;
    else
        *j = 1;

    temp = arrival[*j];
    SelectStream(*j);
    arrival[*j] += Exponential(mean[*j]);

    return temp;
}

double GetService(int j, int n)
{
    const double mean[4] = {1/M1CLET, 1/M2CLET,
                           1/M1CLOUD, 1/M2CLOUD,
                           1/MSETUP};

    SelectStream(j + n + 2);
    return Exponential(mean[j + n]);
}
```

### 4.3 Flusso principale

Il programma che si occupa dell'esecuzione della simulazione è contenuto nel file *cloudq.c*. Il flusso di esecuzione principale consiste nell'eseguire le varie repliche della simulazione producendo, per ognuna di esse, dei file di output che vengono presi in input da altri programmi che si occupano di elaborare i dati, tali programmi sono denominati *bm\_\*.c* e producono, per ogni replicazione e per ogni metrica, un campione di  $k$  medie di batch di dimensione  $b$  sul quale viene calcolato un intervallo di confidenza del 95%.

Ogni replicazione della simulazione è composta dalle seguenti fasi:

1. Inizializzazione: apertura dei file di output, reset delle variabili (e del clock virtuale?), settaggio del seme per il PRNG, generazione ed inserimento nella coda del primo evento di arrivo.
2. Processamento degli eventi: fintanto che la coda degli eventi non è vuota, viene aggiornata l'area relativa alla popolazione nel tempo di simulazione corrente, viene estratto l'evento in cima alla coda ed a seconda del tipo di evento si attuano le azioni descritte nell'algoritmo 1;
3. Terminazione: chiusura dei file di output e stampa su schermo dei risultati della replicazione.

cloudq.c

```

/* initialize data structures */
/* ..... */

while (queue.head != NULL) {

    e = dequeue_event(&queue);
    t.next = e->time;                                /* next event time */

    for (i = 0; i < 5; i++)                            /* update integral */
        area[i] += (t.next - t.current) * n[i];

    t.current = t.next;                                /* advance the clock */

    switch (e->type) {
    case E_ARRIVL:
        /* process an arrival */
        /* ..... */
    case E_SETUP:
        /* process an arrival */
        /* ..... */
    case E_DEPART:
        /* process a departure */
        /* ..... */
        /* write data to outfile */
        /* ..... */
    default:
        handle_error("unknown_event_type");
    }
}

/* ..... */

```

### 4.3.1 Gestione degli eventi

A livello computazionale, le operazioni che corrispondono ai vari eventi (indicati con le macro specificate nel file `basic.h`) sono le seguenti:

- E\_ARRIVL** : evento di arrivo, a seconda della classe del job e dello stato del sistema, vengono aggiornate le variabili degli arrivi e quelle della popolazione corrente, tramite la funzione `srvjob()` viene creato e inserito nella coda un evento di partenza dal nodo in cui il job va concettualmente in esecuzione, per un tempo di servizio che viene generato tramite la funzione `GetService()` e memorizzato nell'apposita variabile del job. Se si verificano le condizioni per cui avviene l'interruzione di un job, tramite la funzione `rplcjob()` viene rimosso un evento di partenza relativo ad un job di classe 2 in esecuzione sul cloudlet, viene creato un evento di setup a cui viene associato il nodo rimosso con un nuovo tempo di servizio generato dalla funzione `GetService()`, infine viene creato ed immesso nella coda un evento di partenza dal cloudlet con associato il nuovo job appena arrivato. Una volta che un evento di arrivo viene processato, viene generato il successivo ed inserito nella coda. Si può osservare che, ad ogni istante, nella coda è presente un solo evento di arrivo, poiché un evento di tale tipo viene generato soltanto dopo che il precedente viene processato.
- E\_DEPART** : evento di partenza, vengono aggiornate le variabili relative alla popolazione del sistema ed ai completamenti in base alla classe del job e al nodo di servizio, inoltre vengono scritti i dati correnti sui file di output, quindi, una metrica di interesse, viene registrata ad ogni completamento di un job, anche quelle non relative ai singoli job come la popolazione media.
- E\_SETUP** : evento di setup, indica la terminazione della fase di setup di un job interrotto, viene generato un nuovo evento di partenza dal cloud e viene aggiornato lo stato della popolazione del sistema.
- E\_IGNRVL** :

[CODICE RELATIVO AI VARI EVENTI]

## 4.4 Produzione ed Elaborazione dell'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 (`bm_*.c`) per il calcolo delle metriche di interesse.

### 4.4.1 Tempo di Servizio

Il file `service.dat` contiene le informazioni riguardanti i tempi di servizio di ogni job in base alla sua classe ed al nodo su cui è stato eseguito: ogni riga corrisponde ad un singolo job tranne la prima in cui sono presenti i completamenti sempre



cloudq.c

```

double srvjob(struct job_t job, unsigned int node,
              struct queue_t *queue, clock t)
{
    double service = GetService(job.class, node);
    struct event *e = alloc_event();
    memset(e, 0, sizeof(struct event));

    job.node = node;
    job.service[job.class + node] = service;
    e->job = job;
    e->time = t.current + service;
    e->type = E_DEPART;
    enqueue_event(e, queue);

    return service;
}

/* return the cloudlet execution time of the removed job */
double rplcjob(struct queue_t *queue, clock t, unsigned int n)
{
    double service;
    double left;
    double setup = GetService(J_CLASS2 + SETUP);
    struct event *temp = alloc_event();
    struct job_t *job = &temp->job;
    struct event *e = NULL;

    job->class = J_CLASS2;
    job->node = CLET;
    temp->type = E_DEPART;
    e = remove_event(queue, temp, rmpos(n));

    left = e->time - t.current;

    e->time = t.current + setup;
    e->type = E_SETUP;

    job = &e->job;
    job->node = SETUP;
    job->service[J_CLASS2 + SETUP] = setup;
    service = job->service[J_CLASS2 + CLET];
    job->service[J_CLASS2 + CLET] -= left;

    enqueue_event(e, queue);
    free(temp);

    return service - left;
}

```

suddivisi per combinazione classe-nodo, necessari al calcolo della grandezza dei batch. Infatti se durante la simulazione sono stati processati  $c_1$  job di classe 1 ed  $c_2$  job di classe 2, la dimensione dei relativi batch sarà rispettivamente  $c_1/K$  e  $c_2/K$ , con  $K$  pari al numero di batch del campione.

Poiché i tempi di servizio vengono scritti ad ogni completamento dei job, quindi in un ordine che non corrisponde a quello di arrivo, è necessario riordinare

le righe del file, questo viene fatto tramite la chiamata di sistema *system()*, che permette di eseguire il comando shell per il riordino delle righe, a tale scopo viene inserito l'id del job (assegnato in ordine di arrivo) all'inizio di ogni riga.

[CODICE SCRITTURA SU FILE E RIORDINO]  
[FIGURA FILE DI OUTPUT]

Il programma *bm\_srv* si occupa dell'elaborazione di questo file, leggendo riga per riga e riempiendo le apposite variabili, distinguendo la tipologia del job ed il nodo di esecuzione in base ai valori non nulli della riga.

Ogni metrica è implementata con un array di  $K$  elementi, i quali vengono popolati sommando esattamente  $b$  valori della metrica che compongono un batch. Al termine del ciclo di processamento delle righe, ogni elemento viene diviso per  $b$  per calcolare il valor medio.

In questo modo l'array costituisce un campione di dimensione  $K$  per la metrica, per il quale il programma calcola un intervallo di confidenza con un livello di significatività pari a  $\alpha = 0.05$ .

[CODICE *bm\_srv.c*]

#### 4.4.2 Throughput, popolazione, e percentuale di interruzioni

I file *throughput.dat*, *population.dat* e *interruption.dat* contengono valori relativi rispettivamente al throughput, alla popolazione media e alla percentuale di job interrotti.

Anche se sono metriche che non si riferiscono direttamente ad un job, è stato comunque scelto l'evento di completamento come istante di campionamento, in modo da avere conformità tra tutti i file di output e controllo sul numero di valori che vengono raccolti, poiché il programma raccoglie una quantità di dati pari al numero di job che devono essere processati durante la simulazione.

I file vengono progressivamente generati, come per il file *service.dat*, scrivendo su ogni riga i valori corrispondenti alla combinazione classe-nodo.

L'elaborazione differisce invece perché ogni riga contribuisce al popolamento degli array, quindi ognuno di questi avrà i batch della stessa dimensione pari al rapporto tra il numero di righe  $N\_JOBS$  e la dimensione del campione  $K$ .

[CODICE ED ESEMPIO DI OUTPUT]  
[CODICE *bm\_thr.c*]

## 5 Modello Analitico

In questa sezione viene mostrata la metodologia utilizzata per la stima delle statistiche. Nel modello analitico le statistiche si basano sui valori medi delle variabili aleatorie  $S$ ,  $N$  e  $X$ , corrispondenti rispettivamente al tempo di risposta, popolazione e throughput.

Per il calcolo delle metriche locali e globali, è stato inoltre necessario ottenere la distribuzione stazionaria dello stato del cloudlet, in modo tale da poter stabilire lo stato di tutto il sistema in condizioni di stazionarietà, e a tale scopo, si è modellato il cloudlet come una catena di markov.

### 5.1 Calcolo della distribuzione stazionaria

Il generico stato della catena di markov è rappresentato da una coppia di interi  $(n_1, n_2)$  che indicano la quantità di job della relativa classe presenti nel cloudlet.

La frequenza di transizione da uno stato all'altro è regolata dai tassi di arrivo  $\lambda_1$  e  $\lambda_2$  e di completamento  $\mu_1^{clet}$  e  $\mu_2^{clet}$  che verranno indicati semplicemente con  $\mu_1$  e  $\mu_2$ , poiché la catena di markov è riferita soltanto al cloudlet.

La catena rispetta i vincoli imposti relativi alla somma delle variabili  $n_1$  e  $n_2$ , gli stati risultanti sono mostrati in figura 3: le frecce tratteggiate indicano il fatto che i job, al loro arrivo, vengono direttamente inoltrati al server remoto, mentre nel caso in cui arrivi un job di classe 1 nello stato  $(n_1, n_2)$  tale che  $n_1 + n_2 = S$ , si passa allo stato  $(n_1 + 1, n_2 - 1)$  e ciò rappresenta una migrazione di un job di classe 2.

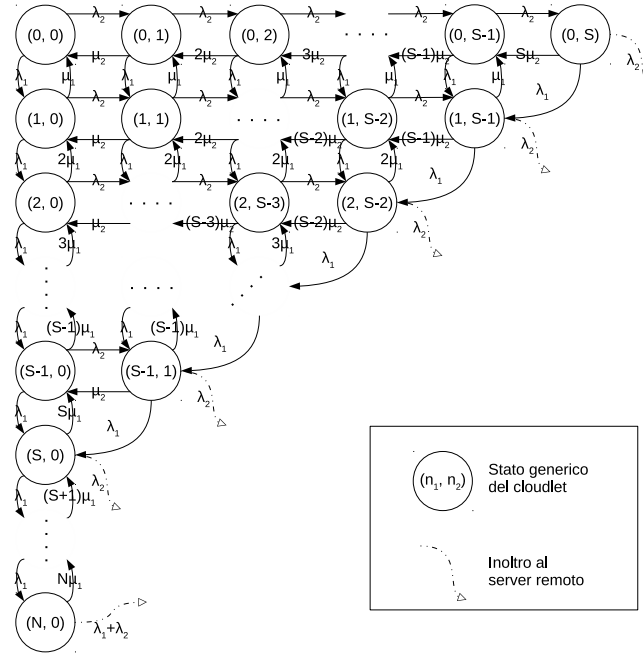


Figura 3: CTMC cloudlet

Per il calcolo della distribuzione stazionaria si è utilizzato un programma di calcolo basato su matrici scritto in *Matlab*, al fine di risolvere il seguente sistema di equazioni di bilancio che deriva dall'analisi della catena.

$$\left\{ \begin{array}{ll}
\lambda\pi_{(0,0)} - \mu_1\pi_{(1,0)} - \mu_2\pi_{(0,1)} = 0 & \\
(\lambda + i\mu_1)\pi_{(i,0)} - \lambda_1\pi_{(i-1,0)} + -(i+1)\mu_1\pi_{(i+1,0)} + & \forall i = 1, \dots, S-1 \\
\quad - \mu_2\pi_{(i,1)} = 0 & \\
(\lambda + j\mu_2)\pi_{(0,j)} - \lambda_2\pi_{(0,j-1)} - \mu_1\pi_{(1,j)} + & \forall j = 1, \dots, S-1 \\
\quad - (j+1)\mu_2\pi_{(0,j+1)} = 0 & \\
(\lambda + i\mu_1 + j\mu_2)\pi_{(i,j)} - \lambda_1\pi_{(i-1,j)} - (i+1)\mu_1\pi_{(i+1,j)} + & \forall i, j = 1, \dots, S-1 : i+j < S \\
\quad - \lambda_2\pi_{(i,j-1)} - (j+1)\mu_2\pi_{(i,j+1)} = 0 & \\
(\lambda_1 + i\mu_1 + j\mu_2)\pi_{(i,j)} - \lambda_1(\pi_{(i-1,j)} + \pi_{(i-1,j+1)}) + & \forall i, j = 1, \dots, S-1 : i+j = S \\
\quad - \lambda_2\pi_{(i,j-1)} = 0 & \\
(\lambda_1 + S\mu_2)\pi_{(0,S)} - \lambda_2\pi_{(0,S-1)} = 0 & \\
(\lambda_1 + S\mu_1)\pi_{(S,0)} - \lambda_1(\pi_{(S-1,0)} + \pi_{(S-1,1)}) + & \\
\quad - (S+1)\mu_1\pi_{(S+1,0)} = 0 & \\
(\lambda_1 + i\mu_1)\pi_{(i,0)} - \lambda_1\pi_{(i-1,0)} - (i+1)\mu_1\pi_{(i+1,0)} = 0 & \forall i = S+1, \dots, N-1 \\
\sum_{(i,j) \in E} \pi_{(i,j)} = 1 &
\end{array} \right.$$

Si è quindi giunti ad ottenere le probabilità  $\pi_{(n_1, n_2)}$  di trovarsi nello stato  $(n_1, n_2) \in E$  in condizioni di stazionarietà, dove  $E$  è l'insieme dei possibili stati della catena.

A questo punto è possibile, sommando le opportune  $\pi_{(n_1, n_2)}$ , ricondursi alle probabilità che si verifichino determinati eventi che consentiranno di stimare le metriche di interesse.

## 5.2 Probabilità preliminari

Prima di tutto è conveniente sfruttare la distribuzione stazionaria per il calcolo di alcune probabilità preliminari, utili al calcolo dei tempi di risposta.

- *Probabilità di Accettazione*: probabilità che il sistema si trovi in uno stato in cui qualunque job viene accettato in esecuzione nel cloudlet

$$\Pi_A = \sum_{\substack{n_1, n_2: \\ n_1 + n_2 < S}} \pi_{(n_1, n_2)} \quad (24)$$

- *Probabilità di Soglia*: probabilità che il sistema si trovi in uno stato in cui il valore di soglia è stato raggiunto

$$\Pi_S = \sum_{\substack{n_1, n_2: \\ n_1 + n_2 \geq S}} \pi_{(n_1, n_2)} \quad (25)$$

- *Probabilità di Blocco*: probabilità che il sistema si trovi in uno stato in cui qualunque job viene direttamente inoltrato al server remoto (cloud)

$$\Pi_B = \sum_{\substack{n_1, n_2: \\ n_1 + n_2 = N}} \pi_{(n_1, n_2)} \quad (26)$$

- *Probabilità di Interruzione*: probabilità che il sistema si trovi in uno stato in cui è possibile che un job venga interrotto

$$\Pi_I = \sum_{\substack{n_1, n_2: \\ n_1 + n_2 = N \\ n_2 > 0}} \pi_{(n_1, n_2)} \quad (27)$$

- *Probabilità di Interruzione a seguito di Accettazione*: probabilità che un job accettato nel cloudlet venga interrotto, calcolata rapportando la frequenza di interruzione di un job alla frequenza di accettazione

$$P_{intr}^{clet} = \frac{\lambda_1 \Pi_I}{\lambda_2 \Pi_A} \quad (28)$$

- *Probabilità di Interruzione di un job di classe 2*: probabilità che un job di classe 2 venga interrotto, ovvero che venga accettato nel cloudlet e interrotto

$$P_{intr} = \Pi_A P_{intr}^{clet} \quad (29)$$

### 5.3 Throughput

Per il calcolo del throughput si può sfruttare sempre la condizione di stabilità del sistema e dei suoi nodi, e poi calcolare il throughput del cloudlet sottraendo dal throughput del sistema quello del cloud, poiché le uscite dal nodo di setup non corrispondono a quelle del sistema.

$$X_j^{cloud} = \lambda_j^{cloud} \quad j = 1, 2 \quad (30)$$

$$X_{cloud} = \lambda_1^{cloud} + \lambda_2^{cloud} \quad (31)$$

$$X^{setup} = \lambda_{setup} \quad (32)$$

$$X_j = \lambda_j \quad j = 1, 2 \quad (33)$$

$$X = \lambda \quad (34)$$

$$X_j^{clet} = X_j - X_j^{cloud} \quad (35)$$

$$X_{clet} = X - X_{cloud} \quad j = 1, 2 \quad (36)$$

### 5.4 Tempo di risposta: Cloudlet

Il tempo medio di risposta di un job di classe 1 per il cloudlet è banalmente il reciproco del tasso di servizio relativo:

$$E[S_1^{clet}] = \frac{1}{\mu_1^{clet}} \quad (37)$$

tale valore è indipendente dal parametro  $S$  ed equivale a  $2.\bar{2}$ , mentre per quanto riguarda i job di classe 2, il tempo medio di servizio subisce una riduzione dovuta alle interruzioni. Infatti, poiché i job che hanno un tempo di servizio maggiore del valor medio ( $1/\mu_2^{clet}$ ) permangono nel nodo per più tempo, sono più soggetti alle interruzioni, quindi non verranno inclusi nel calcolo del tempo medio ed incideranno maggiormente i job con un tempo di servizio minore. Si

può stimare che, tale tempo di risposta medio, viene abbattuto di un fattore  $E[S_r]$  proporzionale alla probabilità che un job venga interrotto.

$$\begin{aligned} E[S_r] &= \frac{1}{\mu_2^{clet}} P_{intr}^{clet} \\ E[S_2^{clet}] &= \frac{1}{\mu_2^{clet}} - E[S_r] \end{aligned} \quad (38)$$

Tale stima è ragionevole, poiché se si ragiona al limite si ha che, in assenza di interruzioni, il tempo di risposta assume il valore normale  $1/\mu_2^{clet}$ , altrimenti, con una probabilità di interruzione pari a 1, assume giustamente il valore nullo, poiché non ci sarebbero completamenti in tali condizioni.

$$\begin{aligned} P_{intr}^{clet} \rightarrow 0 &\quad \Rightarrow \quad E[S_r] \rightarrow 0 ; \quad E[S_2^{clet}] \rightarrow \frac{1}{\mu_2^{clet}} \\ P_{intr}^{clet} \rightarrow 1 &\quad \Rightarrow \quad E[S_r] \rightarrow \frac{1}{\mu_2^{clet}} ; \quad E[S_2^{clet}] \rightarrow 0 \end{aligned}$$

Il tempo di risposta medio globale per il cloudlet è calcolato come una media pesata sul tipo di job che attraversano il nodo. Nello specifico si ha che una porzione pari a  $\lambda_1/(\lambda_1 + \lambda_2)$  degli arrivi totali al cloudlet è di classe 1 e sarà in servizio per un tempo mediamente pari a  $E[S_1^{clet}]$ , la restante porzione ( $\lambda_2/(\lambda_1 + \lambda_2)$ ) invece è di classe 2 e sarà in servizio per un tempo mediamente pari a  $E[S_2^{clet}]$ . Si ottiene quindi la formula seguente:

$$E[S_{clet}] = \frac{\lambda_1}{\lambda_1 + \lambda_2} E[S_1^{clet}] + \frac{\lambda_2}{\lambda_1 + \lambda_2} E[S_2^{clet}] \quad (39)$$

### 5.5 Tempo di risposta: Cloud

Per quanto riguarda il cloud, i tempi di risposta specifici per classe sono molto semplici da stimare, e sono pari al reciproco del rispettivo tasso di servizio:

$$E[S_1^{cloud}] = \frac{1}{\mu_1^{cloud}} \quad (40)$$

$$E[S_2^{cloud}] = \frac{1}{\mu_2^{cloud}} \quad (41)$$

Il tempo di risposta globale per il cloud può essere calcolato in maniera analoga al caso del cloudlet, ad eccezione del fatto che i tassi di arrivo non sono noti, ma possono essere calcolati facilmente perché equivalgono ai tassi di scarto del cloudlet.

I job di classe 1 vengono scartati soltanto nel caso in cui il cloudlet si trova nello stato  $(N, 0)$ , ovvero quando vi sono esattamente N job di classe 1 in esecuzione.

I job di classe 2 vengono scartati nel caso in cui il cloudlet si trova in uno stato in cui si è raggiunto il valore di soglia, oppure nel caso in cui avvenga una sostituzione con uno di classe 1. In conclusione si ottiene:

$$\begin{aligned} \lambda_1^{cloud} &= \Pi_B \lambda_1 & \lambda_2^{cloud} &= (\Pi_S + P_{intr}) \lambda_2 \\ E[S_{cloud}] &= \frac{\lambda_1^{cloud}}{\lambda_1^{cloud} + \lambda_2^{cloud}} E[S_1^{cloud}] + \frac{\lambda_2^{cloud}}{\lambda_1^{cloud} + \lambda_2^{cloud}} E[S_2^{cloud}] \end{aligned} \quad (42)$$

### 5.6 Tempo di risposta: Sistema

Un job di classe 1, al suo arrivo nel sistema, può essere soggetto a due alternative: essere inoltrato nel cloud nel caso in cui il cloudlet sia saturo con probabilità  $\Pi_B$ , oppure essere eseguito nel cloudlet con probabilità  $1 - \Pi_B$ .

$$E[S_1] = (1 - \Pi_B) E[S_1^{clet}] + \Pi_B E[S_1^{cloud}] \quad (43)$$

Prima di calcolare il tempo di risposta medio dei job di classe 2, è necessario calcolare il tempo di risposta medio di un job interrotto.

Tale tempo è composto dalla somma di tre componenti, partendo dall'ultima: il tempo medio di servizio nel nodo cloud pari a  $E[S_{cloud}]$ , il tempo medio di setup pari a  $E[S_{setup}]$  ed infine il tempo medio di esecuzione nel cloudlet prima che il job venga interrotto.

La stima di quest'ultimo può essere effettuata similmente a quella del calcolo del tempo di risposta dei job di classe 2 nel cloudlet. Infatti anche in questo caso è necessario sottrarre un certa quantità al tempo che normalmente avrebbe impiegato in esecuzione se non ci fosse stata interruzione, ovvero  $1/\mu_2^{clet}$ . Sarebbe quindi ideale la relazione del tipo:

$$E[Y] = \frac{1}{\mu_2^{clet}} (1 - \beta P_{intr}^{clet}) \quad 0 < \beta \leq 1$$

perché si vuole che il tempo di esecuzione antecedente all'interruzione (qui indicato per semplicità con  $Y$ ) diminuisca all'aumentare della probabilità di interruzione. Si noti che per  $\beta = 1$  si ottiene esattamente la formula di  $E[S_2^{clet}]$ , in questo caso però, non si vuole che il tempo di esecuzione tenda a 0 se la probabilità di interruzione fosse pari a 1, perché tale situazione non sarebbe realistica, infatti anche se ogni singolo job deve essere interrotto non si può pensare che esso abbia un tempo di esecuzione nel cloudlet pari a 0. Per questo motivo,  $\beta$  funge da fattore di attenuazione che riduce l'incisività della probabilità di interruzione sul calcolo.

Tramite il confronto congiunto con le varie simulazioni al variare del parametro  $S$ , è stato scelto il un valore del fattore  $\beta$  pari a 0.95. Si è giunti quindi alla formula seguente per il calcolo del tempo medio di servizio di un job interrotto:

$$E[S_{intr}] = (1 - \beta P_{intr}^{clet}) \frac{1}{\mu_2^{clet}} + E[S_{setup}] + E[S_{cloud}] \quad \beta = 0.95 \quad (44)$$

Un job di classe 2 invece può avere tre diversi tempi di esecuzione in base ai seguenti casi:

- Il job viene inoltrato direttamente al cloud remoto perché si è raggiunto il valore di soglia, dove viene eseguito per un tempo mediamente pari a  $E[S_2^{cloud}]$ , questo avviene con probabilità  $\Pi_S$ ;
- Il job viene accettato nel cloudlet e non viene interrotto, il suo tempo di esecuzione è mediamente pari a  $E[S_2^{clet}]$ , questo accade con probabilità  $\Pi_A(1 - P_{intr}^{clet})$ ;
- Il job viene eseguito nel cloudlet, viene interrotto ed inoltrato al server remoto in cui viene eseguito dopo una fase di setup, questo avviene con probabilità  $P_{intr}$  ed il tempo di esecuzione è mediamente pari a  $E[S_{intr}]$ ;

Pertanto un job di classe 2 ha un tempo di risposta medio che deriva dalla seguente formula:

$$E[S_2] = \Pi_S E[S_2^{cloud}] + \Pi_A (1 - P_{intr}^{clet}) E[S_2^{clet}] + P_{intr} E[S_{intr}] \quad (45)$$

Il tempo di risposta globale del sistema è calcolato analogamente al caso del cloudlet, con i tempi di servizio riferiti al sistema intero, si ha quindi:

$$E[S] = \frac{\lambda_1}{\lambda_1 + \lambda_2} E[S_1] + \frac{\lambda_2}{\lambda_1 + \lambda_2} E[S_2] \quad (46)$$

### 5.7 Popolazione media: Cloudlet

La popolazione media del cloudlet, sia per classe che globale, viene calcolata sommando il prodotto, relativo ad ogni stato, tra il numero di job presenti in esso per la probabilità di trovarci in condizioni di stazionarietà, ovvero:

$$E[N_1^{clet}] = \sum_{(n_1, n_2) \in E} n_1 \pi_{(n_1, n_2)} \quad (47)$$

$$E[N_2^{clet}] = \sum_{(n_1, n_2) \in E} n_2 \pi_{(n_1, n_2)} \quad (48)$$

$$E[N_{clet}] = \sum_{(n_1, n_2) \in E} (n_1 + n_2) \pi_{(n_1, n_2)} \quad (49)$$

### 5.8 Popolazione media: Cloud

Per il calcolo della popolazione media del cloud si può sfruttare il fatto che il nodo ha risorse virtualmente infinite (utilizzo  $\rho < 1$ ), pertanto si può considerare stabile ed è possibile applicare la legge di Little:

$$E[N_j^{cloud}] = \lambda_j^{cloud} E[S_j^{cloud}] \quad j = 1, 2 \quad (50)$$

$$E[N_{cloud}] = (\lambda_1^{cloud} + \lambda_2^{cloud}) E[S_{cloud}] \quad (51)$$

### 5.9 Popolazione media: Setup

Anche il nodo di setup può essere considerato stabile per gli stessi motivi. Si può applicare la legge di Little con un tasso di arrivi a questo nodo pari al tasso di interruzione dei job di classe 2.

$$\begin{aligned} \lambda_{setup} &= P_{intr} \lambda_2 \\ E[N^{cloud}] &= \lambda_{setup} E[S_{setup}] \end{aligned} \quad (52)$$

### 5.10 Popolazione media: Sistema

Anche il sistema si può considerare stabile, perché globalmente ha anch'esso risorse infinite.

$$E[N_j] = \lambda_j E[S_j] \quad j = 1, 2 \quad (53)$$

$$E[N] = (\lambda_1 + \lambda_2) E[S] \quad (54)$$



## 6 Risultati

La validazione dei modelli appena discussi è stata effettuata confrontando i risultati della simulazione con le stime dedotte dall'analisi, valutando quattro possibili scenari contraddistinti dal valore che assume il parametro  $S$ .

Ognuno di questi scenari è stato simulato facendo processare al programma lo stesso numero di job affinché le statistiche globali del sistema possano essere confrontate. Tale numero è stato stabilito in base alle metriche che andavano valutate ed al numero di job che le riguardava.

Lo scenario con valore di soglia  $S = 5$ , è quello in cui soltanto circa il 2% dei job di classe 2 è processato nel cloudlet e di questi circa il 90% vengono interrotti, ne risulta, quindi che in relazione al numero totale dei job che transitano nel sistema, solamente lo 0.14% sono job di classe 2 processati con successo nel cloudlet e l'1.37% sono job interrotti. In definitiva, al fine di avere una dimensione dei batch soddisfacente per il calcolo delle relative metriche, è stato scelto un numero totale di job pari a 500000. In questo modo si ottiene un numero di job di classe 2 processati con successo nel cloudlet pari a circa 700 e un numero di job interrotti pari a circa 6850, ciò significa che per un valore  $k = 64$  corrispondente al numero di batch, si ottengono batch di dimensione 10 e 107 rispettivamente, che sono sufficienti a determinare intervalli di confidenza, seppur con un ampio margine di errore.

Quello appena descritto è il peggior scenario possibile, in cui si registra il minor numero di job per una determinata metrica, gli altri scenari vengono quindi simulati tutti con un numero totale di job pari a 500000, e a seconda del numero di job che verranno processati nei vari nodi e della loro tipologia, sono risultati intervalli di confidenza più o meno precisi.

Purtroppo è risultato impossibile determinare risultati attendibili per i job di classe 1 che vengono processati nel cloud, poiché il loro numero è, per ogni scenario, talmente esiguo, che per avere un batch di dimensione 10 sarebbero necessari almeno 3 milioni di job in tutto. Tuttavia, con un numero di 500000 job totali, si ottengono delle statistiche globali molto attendibili, infatti, ad esempio per il tempo medio di risposta del sistema, si ottiene una dimensione del batch pari a  $\lfloor \frac{500000}{64} \rfloor = 7812$ .

	$s_1^{clet}$	$s_2^{clet}$	$s_{clet}$
R1	$2.2264 \pm 0.0071$	$2.4014 \pm 0.0119$	$2.2919 \pm 0.0049$
R2	$2.2223 \pm 0.0076$	$2.4231 \pm 0.0110$	$2.2984 \pm 0.0049$
R3	$2.2270 \pm 0.0073$	$2.4130 \pm 0.0133$	$2.2971 \pm 0.0046$
R4	$2.2249 \pm 0.0078$	$2.4084 \pm 0.0106$	$2.2940 \pm 0.0047$
R5	$2.2240 \pm 0.0067$	$2.4162 \pm 0.0117$	$2.2973 \pm 0.0055$
R6	$2.2261 \pm 0.0072$	$2.3995 \pm 0.0110$	$2.2917 \pm 0.0051$
R7	$2.2235 \pm 0.0070$	$2.4099 \pm 0.0107$	$2.2940 \pm 0.0049$
R8	$2.2200 \pm 0.0073$	$2.4047 \pm 0.0141$	$2.2894 \pm 0.0056$
R9	$2.2137 \pm 0.0069$	$2.4172 \pm 0.0119$	$2.2906 \pm 0.0048$
R10	$2.2179 \pm 0.0080$	$2.4096 \pm 0.0092$	$2.2905 \pm 0.0050$
STIMA	2.2222	2.3904	2.3248
MAX ERR	0.0121	0.0436	0.0298

Tabella 4: Confronto tempi risposta cloudlet per  $S = 20$ 

	$s_1^{cloud}$	$s_2^{cloud}$	$s_{cloud}$
R1	$2.2264 \pm 0.0071$	$2.4014 \pm 0.0119$	$2.2919 \pm 0.0049$
R2	$2.2223 \pm 0.0076$	$2.4231 \pm 0.0110$	$2.2984 \pm 0.0049$
R3	$2.2270 \pm 0.0073$	$2.4130 \pm 0.0133$	$2.2971 \pm 0.0046$
R4	$2.2249 \pm 0.0078$	$2.4084 \pm 0.0106$	$2.2940 \pm 0.0047$
R5	$2.2240 \pm 0.0067$	$2.4162 \pm 0.0117$	$2.2973 \pm 0.0055$
R6	$2.2261 \pm 0.0072$	$2.3995 \pm 0.0110$	$2.2917 \pm 0.0051$
R7	$2.2235 \pm 0.0070$	$2.4099 \pm 0.0107$	$2.2940 \pm 0.0049$
R8	$2.2200 \pm 0.0073$	$2.4047 \pm 0.0141$	$2.2894 \pm 0.0056$
R9	$2.2137 \pm 0.0069$	$2.4172 \pm 0.0119$	$2.2906 \pm 0.0048$
R10	$2.2179 \pm 0.0080$	$2.4096 \pm 0.0092$	$2.2905 \pm 0.0050$
STIMA	2.2222	2.3904	2.3248
MAX ERR	0.0121	0.0436	0.0298

Tabella 5: Confronto tempi di risposta cloud per  $S = 20$ 

	$s_1$	$s_2$	$s$
R1	$2.2264 \pm 0.0071$	$2.4014 \pm 0.0119$	$2.2919 \pm 0.0049$
R2	$2.2223 \pm 0.0076$	$2.4231 \pm 0.0110$	$2.2984 \pm 0.0049$
R3	$2.2270 \pm 0.0073$	$2.4130 \pm 0.0133$	$2.2971 \pm 0.0046$
R4	$2.2249 \pm 0.0078$	$2.4084 \pm 0.0106$	$2.2940 \pm 0.0047$
R5	$2.2240 \pm 0.0067$	$2.4162 \pm 0.0117$	$2.2973 \pm 0.0055$
R6	$2.2261 \pm 0.0072$	$2.3995 \pm 0.0110$	$2.2917 \pm 0.0051$
R7	$2.2235 \pm 0.0070$	$2.4099 \pm 0.0107$	$2.2940 \pm 0.0049$
R8	$2.2200 \pm 0.0073$	$2.4047 \pm 0.0141$	$2.2894 \pm 0.0056$
R9	$2.2137 \pm 0.0069$	$2.4172 \pm 0.0119$	$2.2906 \pm 0.0048$
R10	$2.2179 \pm 0.0080$	$2.4096 \pm 0.0092$	$2.2905 \pm 0.0050$
STIMA	2.2222	2.3904	2.3248
MAX ERR	0.0121	0.0436	0.0298

Tabella 6: Confronto tempi di risposta sistema per  $S = 20$

	$s_{setup}$	$s_{intr}$
R1	$2.2264 \pm 0.0071$	$2.4014 \pm 0.0119$
R2	$2.2223 \pm 0.0076$	$2.4231 \pm 0.0110$
R3	$2.2270 \pm 0.0073$	$2.4130 \pm 0.0133$
R4	$2.2249 \pm 0.0078$	$2.4084 \pm 0.0106$
R5	$2.2240 \pm 0.0067$	$2.4162 \pm 0.0117$
R6	$2.2261 \pm 0.0072$	$2.3995 \pm 0.0110$
R7	$2.2235 \pm 0.0070$	$2.4099 \pm 0.0107$
R8	$2.2200 \pm 0.0073$	$2.4047 \pm 0.0141$
R9	$2.2137 \pm 0.0069$	$2.4172 \pm 0.0119$
R10	$2.2179 \pm 0.0080$	$2.4096 \pm 0.0092$
STIMA	2.2222	2.3904
MAX ERR	0.0121	0.0436

Tabella 7: Confronto tempi di risposta setup e interruzioni per  $S = 20$ 

	R1	R2	R3	R4	STIMA	MAX ERR
$n_1^{clet}$	$8.9260 \pm 0.0050$	$8.8170 \pm 0.0130$	$8.9151 \pm 0.0082$	$8.8811 \pm 0.0075$		
$n_2^{clet}$	R1	R2	R3	R4	STIMA	MAX ERR
$n_{clet}$	R1	R2	R3	R4	STIMA	MAX ERR
$n_1^{cloud}$	R1	R2	R3	R4	STIMA	MAX ERR
$n_2^{cloud}$	R1	R2	R3	R4	STIMA	MAX ERR
$n_{cloud}$	R1	R2	R3	R4	STIMA	MAX ERR
$n_{setup}$	R1	R2	R3	R4	STIMA	MAX ERR
$n_1$	R1	R2	R3	R4	STIMA	MAX ERR
$n_2$	R1	R2	R3	R4	STIMA	MAX ERR
$n$	R1	R2	R3	R4	STIMA	MAX ERR

Tabella 8: Risultati popolazione media per  $S = 20$ 

## 6.1 Scenario 1: $S = N = 20$

### 6.1.1 Tempi di risposta

### 6.1.2 Popolazione

### 6.1.3 Throughput

## 6.2 Scenario 2: $S = \frac{N}{2} = 10$

### 6.2.1 Tempi di risposta

### 6.2.2 Popolazione

### 6.2.3 Throughput

## 6.3 Scenario 3: $S = \frac{3}{4}N = 15$

### 6.3.1 Tempi di risposta

### 6.3.2 Popolazione

### 6.3.3 Throughput

## 6.4 Scenario 4: $S = \frac{N}{4} = 5$

### 6.4.1 Tempi di risposta

### 6.4.2 Popolazione

### 6.4.3 Throughput

## 7 Conclusioni

## Riferimenti bibliografici

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