Assignment 6

Salvatore Corvaglia, Franco Savino, Marco Villani

13/12/2017

1 - Building a simple model

Si tratta di progettare un modello che rappresenta una rete composta da due server.

I dati che arrivano al primo server vengono processati uno alla volta e se esso risulta occupato a processare i dati precedentemente arrivati, i nuovi arrivi vengono gestiti tramite una coda o buffer di input. I dati processati vengono poi inviati al secondo server che gestisce lo stesso meccanismo di coda del primo server.

Per implementare questo modello utilizziamo uno strumento di simulazione VIMS, in questo caso usiamo SIMIO, un'applicazione software che ci permette di prendere le migliori decisioni, consentendo di vedere l'impatto delle modifiche proposte prima della loro attuazione. L'obiettivo della simulazione è di imitare il comportamento di un sistema reale che si comporta in modo simile. La simulazione permette di: vedere il futuro, documentare il processo, risposta What If, evitare possibili errori e migliorare le prestazioni.

Utilizzando le librerie che il programma mette a disposizione, si costruisce graficamente il modello sopra descritto usando come nodo di partenza un Source Object che genera un flusso (infinito per default) di specifici tipi di oggetti entità. Mentre il nodo di arrivo viene rappresentato tramite un Sink Object che distrugge gli oggetti entità, i quali sono stati già processati all'interno del modello. I due server sono descritti dai Server Object che modellano un processo. Tramite un path object viene definito il percorso tra due nodi, il tempo di attraversamento da un nodo all'altro dipenderà dalla lunghezza del percorso e dalla velocità delle entità.

2 - Running experiments and viewing results

Sul modello costruito nel primo esempio proviamo a effettuare diversi tipi di esperimenti.
Possiamo impostare sul server1 la capacità di tale risorsa, intesa come il numero di lavori pianificati, e anche il tempo di elaborazione richiesto per processare le entità.

Impostiamo in questo caso il nome del buffer di input di entrambi i server in InputBufferCapacity1 e InputBufferCapacity2. La capacità dei buffers di input definiscono il numero delle entità che possono essere accumulate prima di essere processate.

Successivamente definiamo gli scenari di test (New Experiment), e impostiamo

nella finestra che descrive gli scenari la capacità dei buffer di input dei due server, specificando valori differenti per ogni scenario. In questo caso abbiamo creato tre scenari.

Lanciando il simulatore il modello simulerà gli scenari creati e alla fine dell'esecuzione possiamo visualizzare i dati ottenuti o in delle tabelle di pivot, utili per effettuare un'analisi più dettagliata del sistema, o in un report tradizionale utile per un interesse di tipo statistico.

3 - Modeling material handling systems

Il sistema che abbiamo simulato è composto da due nodi che generano entità che viaggiano con un tasso di velocità differente e attraversano un punto di fusione (cioè un punto in comune dove le entità generate con velocità differenti si incontrano). Esse raggiungono il server e vengono processate una alla volta. Successivamente vengono raccolte e tramite un oggetto trasportatore (Vehicle object) una o più entità (in questo caso abbiamo impostato un massimo di due entità) vengono trasportate (ed eliminate perché già processate) al nodo di arrivo.

Per fare ciò prima si imposta "Interarrival Time" di entrambe le sorgenti, che rappresenta l'intervallo di tempo che intercorre tra due arrivi successivi, tale proprietà viene specificata usando un campione casuale avente una determinata distribuzione di probabilità (in questo caso è esponenziale). Il valore di questo attributo deve essere diverso per entrambe le sorgenti in quanto una genererà entità ad una velocità superiore rispetto all'altra.

Le entità tra le due sorgenti e il server sono trasportate utilizzando un Conveyor object.
Mentre il percorso tra il server e il "depart node" è di tipo bidirezionale, in quanto si sta utilizzando su tale percorso un trasportatore.
Abbiamo impostato anche le velocità dei tre segmenti di nastro trasportatore, inizializzandoli con valori di velocità differenti, espressa in metri al secondo (modificando l'attributo "Initial Desired Speed").

Successivamente abbiamo impostato la capacità del buffer di input del server pari a zero, in modo tale che le entità, una volta raggiunto il server, non vengano accumulate e quindi non dovranno attendere un tempo di attesa prima di essere processate.

Impostiamo la velocità iniziale del veicolo che trasporta le entità una volta processate (tramite l'attributo "Initial Desired Speed", inizializzato con un valore pari a 4 metri al secondo).
Tramite "Initial Node (Home)" settiamo la posizione in cui si deve trovare il veicolo quando viene avviata l'esecuzione della simulazione, in questo caso si deve trovare sul nodo che rappresenta il server. Mentre tramite "idle Action" decidiamo dove si deve trovare nel momento in cui esso non ha nessuna entità

da trasportare, in questo esempio si deve trovare "park at home", dove con home abbiamo indicato il server, come specificato precedentemente. Infine abbiamo impostato tramite l'attributo "Initial Ride Capacity" la capacità iniziale del trasportatore (impostandola a 2 entità).

Il veicolo viene poi animato tramite i seguenti passi:
"Attached queue" → "Queue State" → Output@Server1.ParkingStation.Contents.

Infine per far in modo che le entità non viaggiano singolarmente, ma bensì sul nastro trasportatore si setteranno le seguenti variabili sul nodo di output del server:

"Ride On Transporter" = true
che abilita il trasporto delle entità in uscita al server, e quindi già processate, tramite il trasportatore.

"Transporter Name" = Vehicle 1
Che indica il nome del veicolo incaricato al trasporto delle entità.