REMON – REMOTE MONITOR

V.1.0

Progetto d'esame per il corso di Linguaggi Dinamici
Andrea uguzzoni matricola #56452 AA. 2013/2014
Corso di Laurea in Informatica
Università di Modena
e Reggio Emilia

INDICE:

CAPITOLO 1

DESCRIZIONE DEL PROGETTO

Il progetto nasce da uno spunto introdotto a lezione sulla possibilità di realizzare un sistema di monitoraggio per nodi remoti distribuiti sulla rete locale e dalla grande efficacia dimostrata dallo strumento Monitor di Sistema inserito nelle distribuzioni Debian-like di Linux che organizza le informazioni del consumo di risorse in grafici e li mostra all'utente.

Ho quindi deciso di sviluppare il mio progetto in modo tale da poter mostrare dei grafici in tempo reale dei principali dati di carico di un sistema, avendo però come obiettivo dei nodi locali.

CAPITOLO 2

MODULAZIZZAZIONE

Ho optato per suddividere il progetto in cinque moduli principali per separare adeguatamente i compiti da svolgere, in particolare:

MODULO CONFIGURE

Questo modulo si occupa di gestire i parametri di configurazione del progetto, mostra i valori di default e si occupa del caricamento e salvataggio su file per un successivo riutilizzo.

MODULO SERVER

Questo modulo implementa il server PyRo per gestire le comunicazioni tra i nodi appartenenti allo stesso segmento di LAN, si occupa di fornire indicazioni alle Probe sul funzionamento ad ogni

intervallo di campionamento e fornisce i risultati del monitoraggio al Client, quando questi li richiede. Sempre in questo modulo vengono misurati i tempi di ritardo tra la richiesta di un lavoro e la sottomissione della relativa risposta, al netto del tempo necessario per il campionamento. Si occupa inoltre della gestione delle Probe inteso come avvio ed interruzione del monitoraggio, rilevazione di quelle che sono inattive e mantenimento di uno stato coerente della lista delle Probe cosi che sia analizzabile dal Client.

MODULO PROBE

Questo modulo procede all'effettiva raccolta di dati sul sistema ospite, ad ogni intervallo di campionamento si occupa di raccogliere:

- numero di processori presenti
- percentuale di utilizzo di CPU
- percentuale di utilizzo di Memoria RAM
- percentuale di utilizzo di Memoria Swap

Mentre, previa richiesta da parte del Server può effettuare:

- calcolo del fattoriale di un numero casuale
- allocazione di un numero casuale di locazioni di memoria

Queste due funzionalità servono per mostrare un incremento nel consumo di CPU o RAM nei rispettivi grafici del sistema che ospita la Probe che riceve la richiesta, lanciata attraverso la GUI dall'utente.

MODULO CLIENT

Questo è il modulo che si occupa di analizzare le opzioni in ingresso della chiamata di avvio del programma, ricevere i dati del monitoraggio dal Server, di organizzarli ed effettuare le statistiche per produrre i grafici che saranno mostrati nella GUI e di generare gli stessi.

MODULO INTERFACE

Questo modulo si occupa della gestione dell'interfaccia grafica, caricando il file Glade e legando gli handler alle routine di gestione, gioca un ruolo principale nel progetto perchè è la via di comunicazione principale con l'utente e quindi dialoga con quasi tutti gli altri moduli.

CAPITOLO 3

SCELTE IMPLEMENTATIVE

Ho scelto di fare in modo che il Server venisse richiamato appositamente perchè fosse possibile avviarlo su un qualsiasi nodo della LAN, rimanendo così giustamente slegato dal Client, con il quale può comunicare attraverso PyRo, prevedendo quindi un'opportuna opzione da poter fornire in ingresso al modulo Client, l'elenco completo delle opzioni è:

- -s, --startserver

  alternativa con -g, indica al modulo Client che sul nodo locale si vuole avviare il Server

- -g, --startgui

  alternativa con -s, indica al modulo Client invece che sul nodo locale dovrà partire il Client con la relativa GUI

- -c, --conf NOMEFILE

  indica al modulo Client, quale che sia la modalità richiesta, che si intende utilizzare la configurazione riportata nel file NOMEFILE, se questo è presente e ben formato, altrimenti viene creato il file in questione e vengono scritti i parametri di default

- -l, --localhost

  indica al modulo Client, quale che sia la modalità, che si vuole avviare il programma con funzionamento in locale

- -a, --autostart

  indica al programma di avviare i moduli Server e Client sulla medesima macchina in automatico

Per comodità e chiarezza di gestione ho creato un breve script di Fabric con il compito di gestire i nodi remoti del Laboratorio Base ed è inserito nel percorso extra/fabric

MODULI ESTERNI

Mi sono servito di due moduli esterni più due opzionali per la realizzazione del progetto, entrambi presenti all'indirizzo extra/modules:

- Cairoplot v1.1 - https://launchpad.net/cairoplot
  per la realizzarione dei grafici, l'ho scelto perchè mi è sembrato molto curato ed altamente personalizzabile il risultato finale.
- Lockfile v0.9.1 - https://pypi.python.org/pypi/lockfile
  per la gestione del locking sui file dei grafici, l'ho scelto in quanto risulta immediato nell'utilizzo, garantendo però l'efficacia della procedura di locking.
- Psutil v1.2.1 – OPZIONALE - https://code.google.com/p/psutil/

  utilizzato per la raccolta dei dati di carico del computer ospite, viene caricato il modulo allegato al progetto se non viene trovato nel sistema.

- Fabric v1.8.2 – OPZIONALE - http://docs.fabfile.org/en/1.8/

  utilizzato per la gestione dei nodi locali, nell'invio di file, esecuzione di programmi, pulizia al termine dell'esecuzione. Viene caricato solo nel caso che non si trovi già sul computer ospite.