 RPC - Remote procedure call between different systems

Documentatie

1. Conectarea proceselor din vm-uri diferite la procesul administrator:

* Se creeaza manual unul sau mai multe procese pe un vm ;
* Se stabileste conexiunea intre procesele create si procesul administrator in felul urmator :
* Dupa ce a fost creat procesul acesta va trimite un pachet care contine adresa lui IP (Ip sursa) si IP destinatie, lista cu toate functiile si parametrii lor ;
* Trimiterea acestor informatii se realizeaza printr-un socket pe care il deschide procesul administrator in momentul in care este pornit ;
* Va exista un vector de structura la nivelul procesului administrator unde se vor salva datele primite de la procese ;
* Structura va fi alcatuita dintr-un string - aici se va salva adresa IP a vm-ului procesului si un nod dintr-o lista unde se va salva functia asociata procesului ;

1. Apelare functiilor de catre clienti :

* La nivelul user-ului pentru apelarea functiilor acesta va trimite numele functiei si parametrii specifici acesteia pentru a fi gestionate mai departe de procesul administrator ;
* In spatele apelarii functiei de catre user va sta o librarie standard unde vor fi definite : partea de creare a socket-ului , de conectare, de serializare, de trimitere, de primire, de deserializare si return care se va executa la partea de link-are sau la runtime daca va fi dinamica :
* CREATE socket
* CONNECT
* SERIALIZE
* SEND
* RECV
* DESERIALIZE
* RETURN

1. Componenta procesului administrator:

* Aici va exista o coada de request-uri venite de la clienti si un vector de thread-uri care se vor ocupa de cate un request in parte ;
* Exista un thread principal care asculta socket-ul constant si va popula vectorul de request-uri ;
* Fiecare thread asociat unui element din vector va citi request-ul si in functie de acesta va crea un pachet pe care il va trimite la procesul care are functia ceruta de catre utilizator si la primirea valoarii de return va crea un nou pachet pe care il va trimite utilizatorului ;
* Daca toate thread-urile din vector sunt ocupate cererile vor intra in coada si se va ocupa de ele cate unul primul thread liber;
* Pentru request va fi de tip struct pentru a avea si starea request-ului : running sau nu;

1. Componenta proceselor secundare:

* Aici fiecare process are functia lui. Idei de functii care vor fi implementate :

- Functie hash;

- Determinarea frecventei unui cuvant intr-un fisier;

- Criptarea unor date ;

- Decriptarea unor date;

- Eliminare duplicate ;

- Generare pinuri prin bruteforce

* Vor exista 2 thread-uri la nivelul fiecarui process secundar: unul se va ocupa cu ascultarea request-urilor venite de la procesul administrator de unde va primi paramterii pentru functia acestuia, celalalt thread se va ocupa cu trimiterea rezultatului returnat de functie sub forma unui pachet ;

1. Forma pachetelor ce se vor schimba intre procese :

* Pachetul trimis de procesele secundare pentru conectare catre procesul administrativ :

IP\_Sursa & IP\_Destinatie | Denumire\_Func + Parametrii

* Pachetul trimis de procesul administrativ catre procesul secundar pentru a executa functia ceruta de catre user :

Socket\_S & Socket\_D | Nume\_Funct | Parametrii | Suma\_Control(\*)

* Pachetul trimis de procesul secundar catre procesul administrativ cu rezultatul returnat de functia apelata:

Socket\_S & Socket\_D | Return\_Type | Parametrii | Suma\_Control(\*)