Università degli Studi di Università di Bologna

Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea Triennale in Scienze e Tecnologie Informatiche

Relazione Finale di Tirocinio

Framework per Data Distribution Management

Candidato:
Ricci Nicholas
Matricola 0000624982

Tutor Accademico:
Gabriele D'Angelo

Anno Accademico 25/11/2013–?

Indice

Capitolo 1

Obiettivi

Dato un sistema basilare di comparazione tra algoritmi di DDM creare un framework per poter gestire vecchi e nuovi algoritmi con un input e un output uguale. Il framework dev'essere in grado di:

1. gestire sorgenti di nuovi e vecchi algoritmi;
2. poter comparare nuovi e vecchi algoritmi;
3. restituire in output i risultati in una cartella apposita.

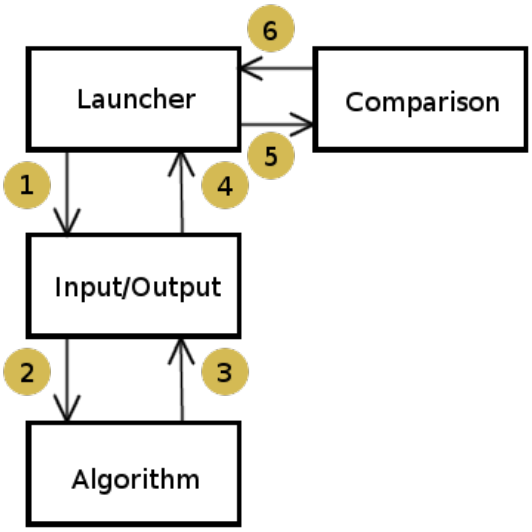

Figura 1.1: Framework schema

Capitolo 2

Cos'è il Data Distribuited Management?

"The goal of HLA Data Distribution Management (DDM) services is to limit the messages received by federates in large distributed federations to those messages of interest in order to reduce (1) the data set required to be processed by the receiving federate and (2) the message traffic over the network. If this functionality is provided in a manner that is efficient, it can significantly improve the performance and scalability of large distributed federations. We provide an overview of the fundamental DDM mechanisms, routing spaces with update and subscription regions, and show how DDM services interoperate with other HLA services to allow federates to discover objects and attributes of interest. We proceed to derive strategies for implementing physically correct filters to account for network latencies and object movement in the virtual environment. These strategies rely on mathematically derived extensions of update and subscription regions to optimize filter efficiency." citazione L. Morse [1]

Capitolo 3

Analisi del sistema

All'interno del sistema di base sono presenti i seguenti oggetti:

1. batch-build_ALL.sh: esegue il comando "make" per gli algoritmi "interval_tree", "sort_matching_standard" e la cartella "utils";
2. batch-clean_ALL.sh: esegue il comando "make clean" per gli algoritmi "interval_tree", "sort_matching_standard" e "utils". Rimuove i file "allspeed*.txt", "*.ps", "*.eps" dalla cartella "_graphs", tutti i file "*.txt" dalla cartella "_result" e tutta la cartella "_bin";
3. batch-graphs_ALL.sh: esegue lo script di shell "batch-speedup.sh" dopodichè sposta i file di risultato nella cartella "_graphs/" e crea i grafici con "gnuplot" per ogni file "*.gp";
4. batch-speedup.sh: Per ogni algoritmo e per ogni alfa, finchè gli "extents" sono minori di "max extents" allora prepara lo speedup per ogni file (algoritmo sequenziale / algoritmo parallelo);
5. configuration.sh: contiene alcune variabili per gli shell script come "start_extents", "max_extents", "step_size", "alfa", "alfas_par", "par_algorithms";
6. batch-tests_ALL.sh: esegue prima lo script "batch-tests_seq.sh" (sequenziale) e successivamente "batch-tests_par.sh" (parallelo)
7. batch-tests_ALL-par.sh: esegue "batch-tests_brute_mp.sh" e "batch-test_interval_mp.sh" in parallelo, con più core;
8. batch-tests_ALL-seq.sh: esegue "batch-tests_brute.sh", "batch-tests_interval.sh" e "batch-tests_sort.sh" in modo sequenziale;
9. batch-tests_brute.sh: crea la cartella "_result" e "_graphs" se non esistono. Per ogni alfa all'interno della variabile ALFAS crea un file

con nome “exec_time_BRUTE_alfa_$ALFA.txt”. Parte da un numero prefissato di extents e finchè non raggiunge un numero massimo di extents esegue 30 run dello stesso algoritmo prendendo così il tempo medio. Ogni tempo medio viene salvato in un file chiamato “exec_time_BRUTE_alfa_$ALFA.txt” e alla fine il file viene spostato nella cartella “_graphs”. Ad ogni extent viene spostato il file “brute_force.txt” nella cartella “_result” che contiene l’esecuzione con quel determinato numero di extents, alfa e dimensione;

10. batch-tests_brute_mp.sh: la stessa cosa di “batch-tests_brute.sh” ma la cartella sorgente è brute_force_mp al posto di brute_force;

11. batch-tests_interval.sh: la stessa cosa di “batch-tests_brute.sh” ma la cartella sorgente è interval al posto di brute_force;

12. batch-tests_interval_mp.sh: la stessa cosa di “batch-tests_brute.sh” ma la cartella sorgente è interval_mp al posto di brute_force;

13. batch-tests_interval_mp_3d.sh: la stessa cosa di “batch-tests_brute.sh” ma la cartella sorgente è interval_mp_3d al posto di brute_force;

14. batch-tests_sort.sh: la stessa cosa di “batch-tests_brute.sh” ma la cartella sorgente è sort al posto di brute_force;

Bibliografia

[1] Katherine L. Morse and Jeffrey S. Steinman. Data distribution management in the hla - multidimensional regions and physically correct filtering. In *In Proceedings of the 1997 Spring Simulation Interoperability Workshop*, pages 343–352. Spring, 1997.