Rularea serviciilor în cadrul unui cluster

Tudor Budu

Rezumat

Scopul temei lucrării prezente este de e a furniza utilizatorilor o modalitate de a rula executabile scrise de aceştia sub un format bine stabilit, pe un cluster de calculatoare scalabil. Astfel, aceste executabile devin servicii pentru utilizatori, existând posibilitatea de a fi rulate în orice moment. Pentru asigurarea unei funcţionări adecvate, cluster-ul trebuie să implementeze mecanisme automate de distribuire a încărcării şi de toleranţă la erori. De asemenea, pentru a asigura securitatea serverului împotriva unor acţiuni ce ar putea compromite buna funcţiune a acestuia, este nevoie de a implementa un sandbox. În cadrul acestuia numărul de operaţii pe care le poate executa un serviciu este limitat doar la cele ce nu implică în nici un mod modificarea stării sistemului, fie prin modificări de fişiere, fie prin apeluri în reţea. Un alt aspect important prezintă posibilitatea de adăugare a unor noduri noi în componenţa cluster-ului fără necesitatea de repornire a acestuia. Aceste noduri trebuie detectate în mod automat şi folosite în momentul potrivit. Iar, pentru a duce cont de acţiunile executate, implementarea unor mecanisme de audit este de asemenea un bonus.

Pentru a satisface cât mai bine aceste cerinţe, o arhitectură potrivită reprezintă o necesitate. Astfel, a fost aleasă o abordare mixtă între arhitectura bazată pe straturi şi cea bazată pe evenimente. Stratul Web furnizează o interfaţă către exterior şi se foloseşte de stratul de acces la baza de date, doar citind date din acesta. În cazul în care cererea implică o oarecare modificare a bazei de date, este creat un eveniment ce este plasat într-o coadă de evenimente de acelaşi tip. La celălalt capăt, un lucrător ascultă aceste evenimente şi le tratează într-un mod corespunzător. Astfel, sunt utilizate părţile bune ale ambelor arhitecturi, existând posibilitatea de a împărţi cozile de mesaje între mai multe noduri de procesare. De asemenea, permite ca acţiunile de citire să fie executate cât mai rapid, deoarece stratul web comunică direct cu stratul de acces la baza de date.

Pentru implementare, luând în considerare suita de funcţionalităţi, a fost aleasă platforma .NET pentru implementarea serverului de aplicaţii şi platforma AngularJS pentru implementarea aplicaţiei cu care va interacţiona utilizatorul. Astfel, ambele pot fi găzduite pe maşini diferite, divizând atribuţiile între acestea şi mărind gradul de performanţă. Pentru baza de date a fost ales SQL Server din motivul unei integrări bune a acestuia cu .NET. Însă, pentru implementarea cozilor de mesaje în locul MSMQ a fost ales RabbitMQ, atât datorită performanţelor sale mai ridicate, dar şi datorită caracterului său multiplatformă. Astfel, serverul de gestiune a cozilor de mesaje poate fi găzduit pe o maşină rulând sistemul de operare Linux, existând posibilitatea de a implementa politici de securitate mai puternice faţă de Windows.

În timpul implementării au fost folosite principiile programării orientate pe obiect şi a dezvoltării orientată pe teste, calitatea codului fiind o prioritate. Totuşi au existat o serie de probleme, multe dintre care au fost neprevăzute, afectând mult atingerea completă a obiectivelor setate iniţial. Din acest motiv, acestea au fost implementate doar până la o stare satisfăcătoare, lăsând o serie de îmbunătăţiri ce ar putea fi aduse aplicaţiei.