

Quality Research by Using Performance Evaluation Metrics for Software Systems and Components

Asist. Ion BULIGIU, prof.dr. Georgeta ȘOAVĂ

Facultatea de Economie și Administrarea Afacerilor, Universitatea din Craiova

Software performance and evaluation have four basic needs: (1) well-defined performance testing strategy, requirements, and focuses, (2) correct and effective performance evaluation models, (3) well-defined performance metrics, and (4) cost-effective performance testing and evaluation tools and techniques. This chapter first introduced a performance test process and discusses the performance testing objectives and focus areas. Then, it summarized the basic challenges and issues on performance testing and evaluation of component based programs and components. Next, this chapter presented different types of performance metrics for software components and systems, including processing speed, utilization, throughput, reliability, availability, and scalability metrics. Most of the performance metrics covered here can be considered as the application of existing metrics to software components. New performance metrics are needed to support the performance evaluation of component based programs.

Keywords: metrics, software performance, testing, evaluation, reliability, scalability

In timp, multe metrice privind performanța au fost concepute pentru a măsura performanța componentelor software sau a sistemelor software, astfel se poate face clasificare a acestora astfel: clasa metricilor de evaluare privind viteza de procesare și timpul de răspuns; clasa metricilor ce studiază fluxurile de transfer din sistem și între componentele sale, cum ar fi rata de procesare a mesajelor intrate, a cererilor și a tranzacțiilor; următoarea clasă se referă la metricile referitoare siguranța sistemelor, urmată de clasa metricilor privind disponibilitatea sistemelor și în final clasa metricilor de scalabilitate, folosite în analiza și prognozarea îmbunătățirilor în sensul asigurării performanțelor privind viteza de procesare.

Metrice de utilizare

Metricile privind utilizarea sunt concepute să măsoare utilizarea resurselor hardware de către o componentă de sistem (sau un sistem). Aplicațiile software utilizează o mare varietate de tipuri de resurse de sistem, de obicei atenția concentrându-se asupra lățimii de bandă a rețelei, viteza procesorului, memorie cache sau memoria fizică pe disc.

Figura 1. a) prezintă rezultatele testării performanței pentru utilizarea rețelei privind

managementul transferurilor electronice online cu clienții, sistem ale cărui tranzacții sunt într-un număr situat sub 7500. În acest caz, metrica utilizării rețelei este definită de raportul dintre traficul rețelei în Kbps și numărul de agenți între care se desfășoară tranzacțiile, pe parcursul unui interval de timp pentru efectuarea testului de performanță, urmărindu-se protocoalele folosite pentru tranzacții.

În figura 2. b) este prezentată utilizarea sistemului pentru un server single-threaded în termenii utilizării hard-diskului, procesor, memorie RAM și cererile de date pentru diferite procese. Utilizarea pentru o stație de tip client poate fi analizată folosind același set de metrice.

Metrice privind viteza de procesare

Pentru a măsura viteza de procesare a unei componente sau a unui sistem, se pot defini o varietate de metrice legate de viteză. Una din cele mai utilizate metrice este timpul de răspuns al sistemului la comenzile utilizatorului. Această metrice poate fi utilizată în măsurarea mediei, minimului și maximului timpului de răspuns pentru diferite grupuri de utilizatori din sistem. Figura 2. a) reprezintă timpurile de răspuns pentru patru grupuri de utili-

zatori într-un sistem de management al transferurilor electronice on-line cu clienții, bazat pe acces concurrent al utilizatorilor.

Figura 2. b) urmărește timpul de răspuns la diferite comenzi ale sistemului în corelație cu numărul de utilizatorilor cu acces concurrent.

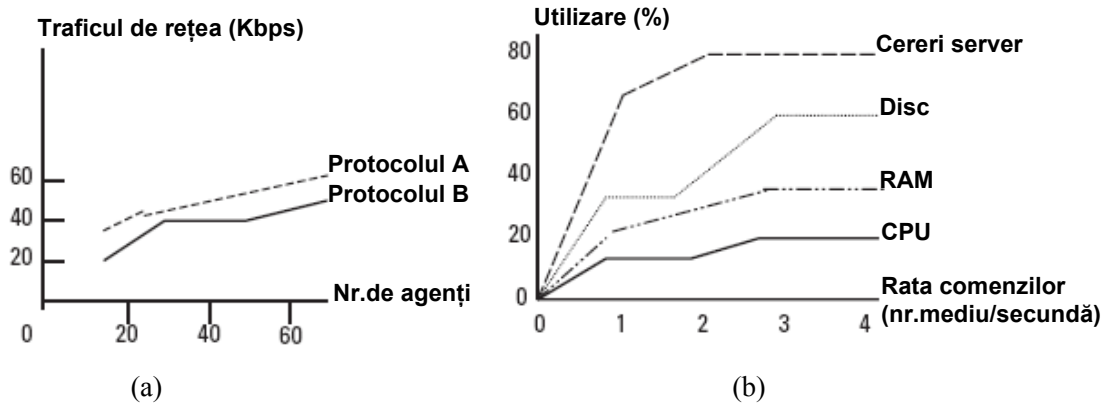


Fig.1. Exemple privind utilizarea sistemelor:

a) pentru o rețea cu mai puțin de 7500 tranzacții; b) pentru un server single-threaded

Pentru componente și sisteme care utilizează funcții cu mesaje de comunicații, se vor analiza timpurile de procesare ale diferitelor tipuri de mesaje. Pentru componente și sisteme cu acces la baze de date, se vor studia timpul de răspuns pentru diferite interogări ale bazelor de date, ținând cont și de timpurile de conectare la bazele de date. Rezultatele obținute referi-

toare la interogarea bazelor de date sunt utile în asigurarea performanței pentru a îmbunătățirea vitezei de citire a datelor prin optimizarea interogărilor.

Figura 2. d) oferă un exemplu în care timpul de răspuns al interogării este măsurat în concordanță cu rata de cererilor efectuate.

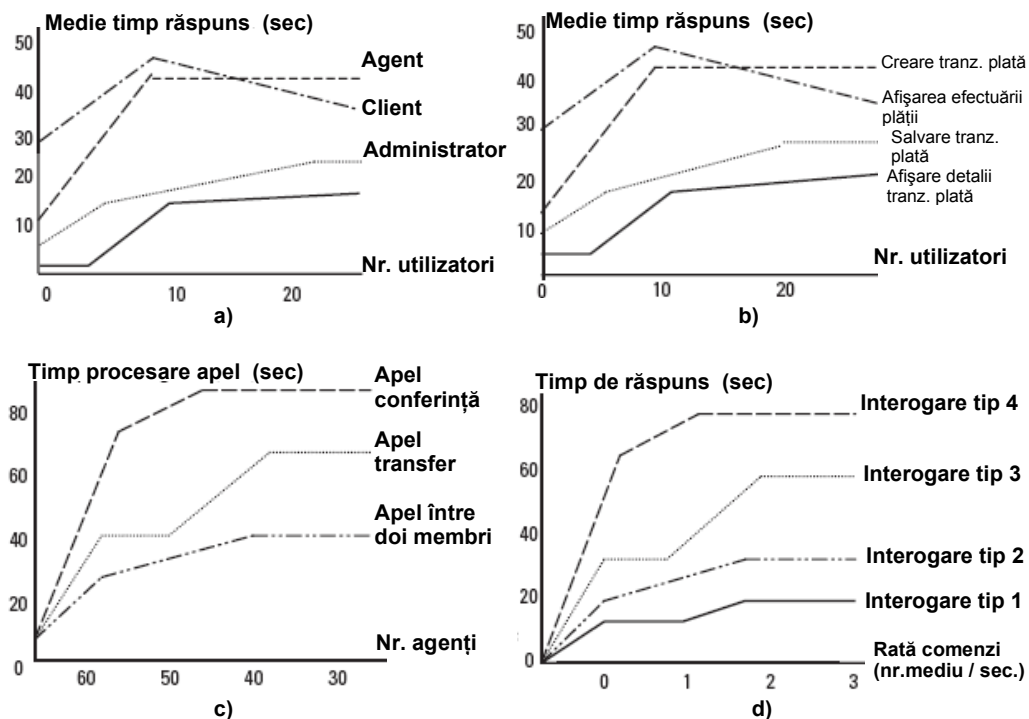


Fig.2. Exemplificări privind viteza unui sistem și a timpului de răspuns:

a) timp de răspuns al unui sistem utilizator; b) timp de răspuns al unui sistem bazat pe comenzi; c) timp de procesare a apelurilor (nr. apeluri 5000/min.) și d) timpuri de răspuns pentru interogări

Măsurarea vitezei pentru un domeniu specific este necesară definirea unor metrici speciale pentru fiecare funcție a domeniului. De exemplu, un apel de procesare este o funcție specifică domeniului pentru un apel al serverului într-un sistem al managementului relațiilor cu clienții. Pentru a măsura viteza de procesare pentru diferite tipuri de apeluri, trebuie calculat timpul de procesare pentru fiecare apel pe baza numărului de agenți disponibili care primesc un tip apel specific (Figura. 2. c.).

Un alt tip de metrici pentru măsurarea vitezei sunt metricile de latență, care măsoară timpii de așteptare sau de întârziere pentru un sistem sau o componentă.

Metriци de disponibilitate

Disponibilitatea componentelor sau a unui sistem este o preocupare generală în cadrul evaluării performanțelor, iar prin intermediul metricilor de disponibilitate a sistemului sau componentelor, acest lucru poate fi studiat prin raportarea timpului în care sistemul este disponibil la timpul total de evaluare a sistemului, ce cuprinde atât timpii în care sistemul este solicitat, cât și timpii în care sistemul este disponibil. Bazându-ne pe această metrică, indisponibilitatea sistemului se calculează ca fiind $1 - \text{disponibilitate}$. Disponibilitatea sistemului poate fi calculată prin relația:

$$\text{disponibilitatea_sistemului} = \frac{\text{timp_sistem_disponibil}}{\text{timp_evaluare_sistem}}$$

unde $\text{timp_sistem_disponibil}$ se referă la timpii în care sistemul este capabil să furnizeze

$$\text{Disponibilitatea_Componentei} = \sum_{j=1}^m \text{Disponibilitate_F_Componenta}(C_k, S, T_p) / m$$

Disponibilitatea extinsă a componentelor

Pentru a testa componentele cu cerințe pentru o disponibilitate mare, trebuie înțeles ce reprezintă o componentă cu disponibilitate extinsă.

O componentă cu disponibilitate extinsă este formată N clustere formate din N componente de redundanță. Ele sunt active în același

funcții și servicii către utilizatori pe toată perioada evaluării.

Ideea de bază poate fi aplicată pentru măsurarea disponibilității sistemului atâta timp componentele software sunt considerate „cutii negre”. Metrica de disponibilitate are două neajunsuri: primul, că nu este prezentă nici o relație între disponibilitatea componentelor și funcțiile suportate de componente, iar cel de-al doilea, nu poate fi aplicată componentelor care au o disponibilitate foarte mare, cum ar fi cele de tratarea erorilor.

Disponibilitatea componentelor bazate pe funcții

Aceste metrici sunt concepute pentru a măsura disponibilitatea componentelor care înglobează funcții.

Disponibilitatea bazată pe funcții a unei componente C_k dintr-un sistem S se calculează prin raportarea timpului total de disponibilitate $T_D(C_k, F_j)$ a funcției F_j în timpul T_p al evaluării performanței. T_p include timpii în care componenta C_k este disponibilă și timpii de indisponibilitate pentru apelul funcției F_j . Formal, disponibilitatea componentei C_k poate fi descrisă prin următoarea relație:

$$\text{Disponibilitate_F_Componenta}(C_k, S, T_p) = \frac{T_D(C_k, F_j)}{T_p}$$

Atâta timp cât această metrică prezintă relația directă între disponibilitatea componentei și funcțiile suportate, măsurarea disponibilității componentei poate fi măsurată prin testarea tuturor funcțiilor componentei pe timpul evaluării performanței.

Pentru o componentă care suportă un număr de m funcții diferite, atunci disponibilitatea ei poate fi calculată astfel:

timp pentru a suporta și a furniza același set de caracteristici ca o „cutie neagră”.

Se consideră $C_{DE} = \{C_0, C_1, \dots, C_N\}$ este un cluster cu N componente de redundanță. Disponibilitatea unei N -cluster componente C_{DE} dintr-un sistem S este cunoscută ca fiind disponibilitate extinsă a componentei. Cuantificarea ei se va face astfel:

$$\text{Disponibilitate_Extinsa_Componenta} = (C_{DE}, S, T_p) = \frac{\text{Disp_Ext} - \text{timp}(C_{DE})}{[\text{Disp_Ext} - \text{timp}(C_{DE}) + \text{Indisp_Ext} - \text{timp}(C_{DE})]}$$

unde $Disp_Ext - timp(C_{DE})$ reprezintă valoarea de timp în care componenta C_{DE} este disponibilă, $Indisp_Ext - timp(C_{DE})$ este timpul în care C_{DE} nu este disponibilă, iar T_p reprezintă perioada de timp pentru evaluarea performanței.

$Disp_Ext - timp(C_{DE})$ se calculează astfel:

$$Disp_Ext - timp(C_{DE}) = \sum_{j=1}^m T_{j(DE)},$$

unde $T_{1(DE)}, T_{2(DE)}, \dots$ și $T_{m(DE)}$ sunt intervale de timp de disponibilitate din timpul total T_p .

În timpul fiecărui interval de timp de disponibilitate, cel puțin o componentă din C_{DE} es-

te activă și disponibilă pentru a pune la dispoziție toate funcțiile componente. Similar se calculează și $Indisp_Ext - timp(C_{DE})$:

$$Indisp_Ext - timp(C_{DE}) = \sum_{j=1}^n T'_{j(DE)}$$

unde $T'_{1(DE)}, T'_{2(DE)}, \dots$ și $T'_{m(DE)}$ sunt intervalele de indisponibilitate din T_p .

În timpul fiecărui interval de timp, toate elementele componente C_{DE} nu sunt disponibile să ofere toate funcțiile. Figura 3 reprezintă un exemplu de trei clustere a unei componente cu disponibilitate extinsă.

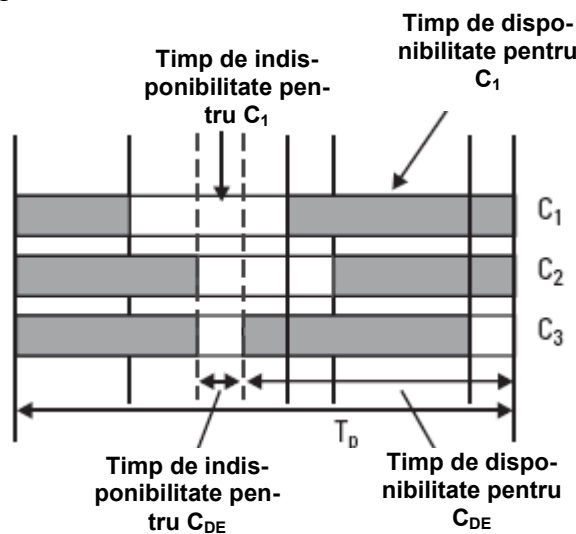


Fig.3. Componentă cu disponibilitate extinsă (C_{DE}) cu trei componente și timpii de disponibilitate și indisponibilitate

Metrici de siguranță

Pentru evaluarea siguranței în funcționare a sistemului se studiază siguranța serviciilor, folosindu-se funcția $R(t)$ ce returnează probabilitatea ca un serviciu să funcționeze într-un interval de timp t .

Siguranța serviciilor este des caracterizată prin specificarea *timpului mediu de defecțiuni* (TMD) sau *a timpului mediu între defecțiuni* (TMID). În calculul TMD sau TMID se ia în considerare faptul că distribuția exponențială descrie cel mai bine posibilele apariții ale defectelor serviciilor.

Siguranța în funcționare a unei componente C_k din sistemul S se referă la raportul dintre timpul total de funcționare al componente și timpul total de evaluare T_p , care include atât

timpii de funcționare, cât și timpii de nefuncționare.

Plecând de la această definiție, siguranța în funcționare a componentelor poate fi ușor evaluată dacă se aplică o componentă de urmărire și înregistrarea a timpilor de funcționare și de nefuncționare.

Dacă se presupune că C_k este o componentă care nu are o disponibilitate extinsă, atunci siguranța în funcționare pe perioada T_p poate fi calculată astfel:

$$Sigur_Comp_{T_p}(C_k) = \frac{timp_func(C_k)}{|T_p|}$$

unde $|T_p|$ reprezintă timpul total de evaluare a performanței, $timp_func(C_k)$ este timpul de funcționare a componente C_k , iar $timp_nfunc(C_k)$ cumulează timpii de nefuncționare și reparare a componente.

Pentru componentele cu disponibilitate extinsă, cum ar fi componentele de tip N-cluster, este nevoie de o altă metrică de evaluarea siguranței în funcționare.

Se va studia mai întâi cazul în care se ia în considerare *criteriul bazat pe apariția unei singure defecțiuni*. Conform acestui criteriu, componenta C_{DE} este considerată nefuncțională dacă un singur element al acestei componente are o defecțiune în orice interval de timp pe parcursul perioadei de evaluare.

Bazându-ne pe această idee, siguranța în funcționare a componente C_{DE} poate fi cuantificată prin intermediul relației:

$Sigur_{CompT_p}(C_{DE}) = \frac{timp_func(C_{DE})}{timp_func(C_{DE}) + timp_nfunc(C_{DE}) / |T_p|}$, unde, $timp_func(C_{DE})$ este timpul de funcționare a componente C_{DE} , iar $timp_nfunc(C_{DE})$ cumulează timpii de nefuncționare și reparare a componente.

Pentru a măsura $timp_nfunc(C_{DE})$, se va presupune că $C_{DE} = \{C_{HA1}, \dots, C_{HAN}\}$ este o componentă de tip N-cluster. În conformitate cu acest criteriu al unei singure defecțiuni, timpul de nefuncționare al C_{DE} poate fi calculat astfel:

$$timp_nfunc(C_{DE}) = \sum_{j=1}^m T_j,$$

unde T_j este un interval de timp din T_p , și cel puțin una din componentele C_{DE} este nefuncțională un timp T_j .

Timpul de funcționare al C_{DE} este calculat prin diferență din timpul total:

$$timp_func(C_{DE}) = |T_p| - [timp_nfunc(C_{DE})].$$

Celălalt criteriu evaluează siguranța în funcționare a componente C_{DE} bazată pe disponibilitatea pentru serviciile funcționale. Prin această abordare, o defecțiune a serviciilor componente C_{DE} pe timpul unui interval de timp presupune că toate componentele redundante ale C_{DE} au eșuat în furnizarea serviciilor lor. Astfel, se utilizează o cale diferită de a calcula timpul de funcționare al componente C_{DE} . Pentru o componentă de tip N-cluster $C_{DE} = \{C_{HA1}, \dots, C_{HAN}\}$, $timp_func(C_{DE})$ se va calcula astfel:

$$timp_func(C_{DE}) = \sum_{j=1}^m T_j,$$

unde T_j este un interval de timp din T_p , și cel puțin una din componentele C_{DE} este funcțională și poate furniza servicii funcționale în timpul T_j .

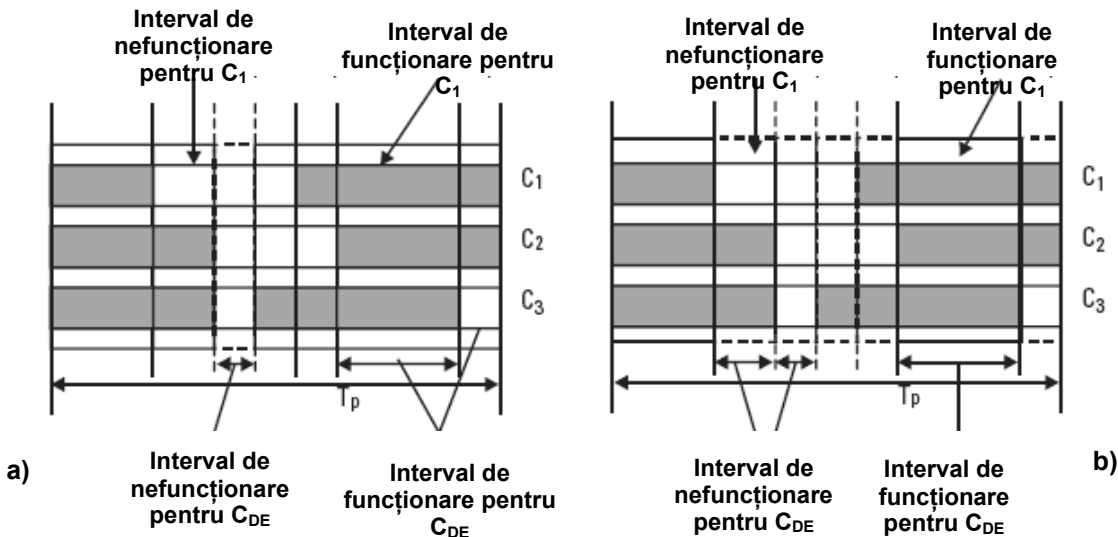


Fig.4. Comparația între cele două criterii privind siguranța în funcționare

Metriци privind rezultatele

Acest tip de metriци este conceput pentru a măsura rata de succes a unei componente software în procesarea evenimentelor, mesajelor și cererilor de tranzacții pe parcursul unei perioade de timp în care se face evaluarea.

Rezultatul tranzacției a unei componente C_k pentru o tranzacție tip TR se referă la numărul total al cererilor de tranzacții procesate cu succes în timpul perioadei de evaluare T_p :

$$rez_tranz(C_k, T_p, TR) = m$$

unde m este numărul total al cererilor de tranzacții procesate cu succes.

Pe parcursul perioadei de testare și evaluare, se pot aplica o serie de teste pentru determinarea maximului, minimului și mediei pentru rezultatele tranzacțiilor la fiecare tip de tranzacție.

Bazându-ne pe metrica de rezultat se poate defini rata rezultatului tranzacției a unei componente C_k ca fiind raportul dintre numărul total de cereri de tranzacții efectuate cu succes și numărul total de cereri de tranzacții primite pe perioada de evaluare T_p :

$$\text{rata_rez}(C_k, T_p, TR) = m / n$$

unde m este numărul total al cererilor de tranzacții TR procesate cu succes în timpul T_p , iar n este numărul total al cererilor de tranzacții TR primite în aceeași perioadă de timp.

Cu cât rata de rezultat se apropie de valoarea 1, componenta C_k dispune de o rată de succes maximă în procesarea cererilor de tranzacții de categoria TR.

Metrici de scalabilitate

Un sistem sau componentă se consideră scalabilă dacă poate fi dezvoltată astfel încât să furnizeze eficient funcții și servicii într-un domeniu predefinit de șabloane. Validarea scalabilității și evaluarea unui sistem și a componentelor sale se face pentru studiul:

- limitelor componentelor sau sistemului în procesare prin prisma dimensiunii problemelor, volumului de date și a numărului de utilizatori cu acces concurent;
- creșterii performanțelor și degradărilor în cazul diferitelor configurații și setări.

Funcția marginală a unei componente

Pentru a măsura limitele performanței unei componente (C_k) trebuie observate cantitativ modificările performanței în execuție pentru o funcție de dimensiune fixă în cazul creșterii firelor de execuție concurente din C_k .

Aplicând teorema Amdahl privind performanța componentelor software și anume că viteza întregului sistem este influențată de cea mai lentă componentă, se va defini timpul de execuție al unei funcții F_j din cadrul unei componente C_k astfel:

$$T(C_k, F_j, n) = T_{\text{serial}} + T_{\text{conc}} / n$$

unde timpul de execuție $T(C_k, F_j, n)$ este împărțit în două: timpul consumat de elementele seriale din C_k , notat cu T_{serial} și timpul de execuție al n diferite fire de execuție concurente, notat cu T_{conc} . Pe măsură ce numărul de fire de execuție concurente crește, cu atât valoarea timpului total de execuție de apropiere de valoarea T_{serial} . În practică, timpul total de execuție tinde către valoarea maximă (numit *punct de prag*) pe măsură ce sunt apar fire de execuție concurente în apelurile componentei C_k . Apariția de noi fire concurente, va solicita și mai mult sistemul prin suprasolicitare, ceea ce va influența și timpul de execuție al componentelor seriale, în sensul creșterii.

Se observă că această metrică poate fi folosită în măsurarea accelerației pentru timpul de execuție al funcției F_j când numărul firelor de execuție concurente al componentei C_k crește de la n la m :

$$T_{\text{accel}}^{F_j}(n, m) = T(C_k, F_j, n) - T(C_k, F_j, m)$$

Valorile negative ale accelerației, desigur, indică o încetinire a execuției, valorile pozitive reprezentând o execuție mai rapidă.

De asemenea, metrica poate fi folosită în determinarea numărului optim de fire concurente pentru a obține o viteză de execuție maximă, însă cu trei limitări:

Limitele rezultatelor unei tranzacții și îmbunătățirea clusterelor componentei

Pentru a crește capacitatea unui sistem de aplicație, de obicei se utilizează componente de timp N-cluster (care reprezintă o grupare de componente identice) care suportă un set specific de funcții. Astfel evaluarea modificărilor privind rezultatele unei procesări pentru o tranzacție se face prin determinarea plașelor de valori între care se situează nivelul rezultatelor și a punctelor de prag la modificarea numărului de componente ale componentei de tip N-cluster.

Se va folosi o metrică simplă pentru observarea modificărilor rezultatelor sistemului pentru tranzacții de tip TR pe măsură ce se modifică numărul componentelor redundante din componenta cluster. Astfel se va măsura rezultatul sistemului în urma unui test fixat T_{set} într-o perioadă dată de timp T_p :

Rezultat_{sistem}(T_{set} , n , TR, T_p) = Numărul total de cereri tranzacții TR procesate cu succes

unde n este numărul componentelor redundante în componenta cluster.

Se deduce că rezultatele sistemului sunt influențate de variația lui n , astfel se pot cuantifica îmbunătățirile aplicate clusterelor componente:

$$Rezultat_{sistem_imb}(n, m) = Rezultat_{sistem}(T_{set}, m, TR, T_p) - Rezultat_{sistem}(T_{set}, n, TR, T_p)$$

unde $Rezultat_{sistem_imb}(n, m)$ reprezintă îmbunătățirea rezultatelor atunci când numărul componentelor identice de tip cluster crește de la n la m .

Bibliografie:

- Cheung, R. C., *A User-Oriented Software Reliability Model*, IEEE Trans. On Software Engineering, Vol. 6, No. 2, March 1980;
- Everett, W. W., "Software Component Reliability Analysis," Proc. of IEEE Symposium on Application-Specific Systems and Software Engineering and Technology, 1999;
- Gao, J., and C. Sun, *Performance Measurement of Software Components—A Systematic Approach*, Technical Report, San José State University, San José, CA, 2002;
- Gao, J., Tsao, H. S. *Testing and quality assurance for component-based software* Artech House, 2003.
- Yacoub, S. M., B. Cukic, and H. H. Ammar, *A Component-Based Approach to Reliability Analysis of Distributed Systems*, Proc. of IEEE 9th International Symposium on Software Reliability Engineering, Paderborn, Germany, 1998;
- Yacoub, S. M., B. Cuki, and H. H. Ammar, *Scenario-Based Reliability Analysis of Component-Based Software*, IEEE 10th International Symposium on Software Reliability Engineering, Boca Raton, FL, 1999;