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

Asist. Ion BULIGIU, prof.dr. Georgeta ŞOAVĂ
Facultatea de Economie şi 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, scalab*ility

În timp, multe metrici 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 şi timpul de răspuns; clasa metricilor ce studiază fluxurile de transfer din sistem şi între componentele sale, cum ar fi rata de procesare a mesajelor intrate, a cererilor şi a tranzacțiilor; următoarea clasă se referă la metricile referitoare siguranța sistemelor, urmată de clasa metricilor privind disponibilitatea sistemelor şi în final clasa metricilor de scalabilitate, folosite în analiza şi prognozarea îmbunătățirilor în sensul asigurării performanțelor privind viteza de procesare.

Metrici de utilizare

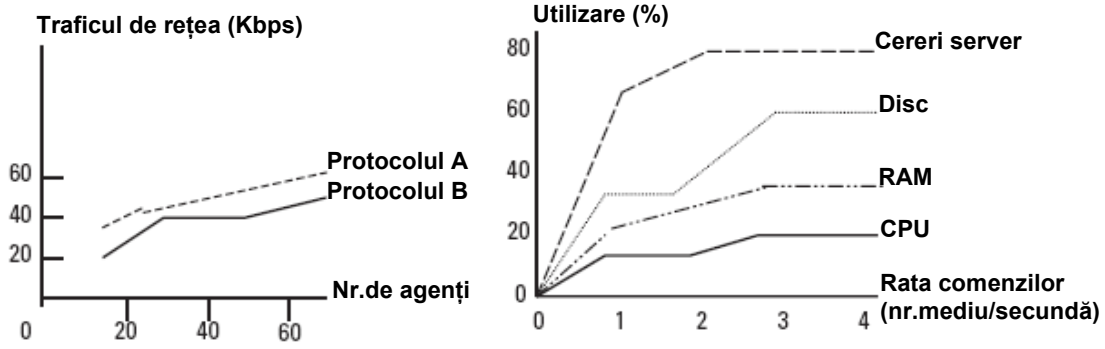
Metricile privind utilizarea sunt concepute să măsoare utilizarea resurselor hardware de către o componentă de sistem (sau un sistem). Aplicațiile software utilizează o mare varietate de tipuri de resurse de sistem, de obicei atenția concentrându-se asupra lățimii de bandă a rețelei, viteza procesorului, 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 şi numărul de agenți între care se desfăşoară 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 şi cererile de date pentru diferite procese. Utilizarea pentru o stație de tip client poate fi analizată folosind acelaşi set de metrici.

Metrici privind viteza de procesare

Pentru a măsura viteza de procesare a unei componente sau a unui sistem, se pot defini o varietate de metrici legate de viteză. Una din cele mai utilizate metrici este timpul de răspuns al sistemului la comenzile utilizatorului. Această metrică poate fi utilizată în măsurarea mediei, minimului şi 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 concurent al utilizatorilor.

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



(a) (b)

Fig.1. Exemple privind utilizarea sistemelor:
a) pentru o rețea cu mai puțin de 7500 tranzacții; b) pentru un server single-threaded

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

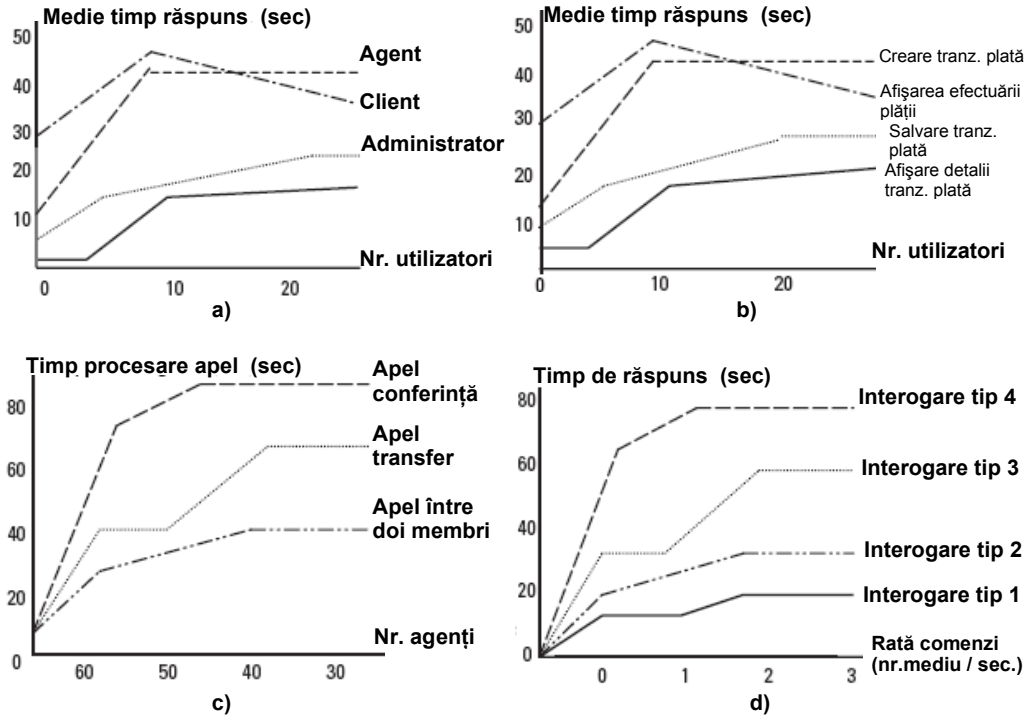


Fig.2. Exemplificări privind viteza unui sistem şi a timpului de răspuns:
a) timp de răspuns al unui sistem utilizator; b) timp de răspuns al unui sistem bazat pe comenzi; c) timp de procesare a apelurilor (nr. apeluri 5000/min.) şi d) timpi de răspuns pentru interogări

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 aşteptare sau de întârziere pentru un sistem sau o componentă.

Metrici 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 şi timpii în care sistemul este disponibil. Bazându-ne pe această metrică, indisponibilitatea sistemului se calculează ca fiind *1 – disponibilitate*. Disponibilitatea sistemului poate fi calculată prin relația:

disponibilitatea_sistemului = timp_sistem_disponibil / timp_evaluare_sistem ,

unde *timp_sistem_disponibil* se referă la timpii în care sistemul este capabil să furnizeze funcții şi 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 şi 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 şi timpii de indisponibilitate pentru apelul funcției F_j. Formal, disponibilitatea componentei C_k poate fi descrisă prin următoarea relație:

$$Disponibilitate\_F\_Componenta\ (C_k, S, T_P) = T_D(C_k, F_j) / T_P$$

Atâta timp cât această metrică prezintă relația directă între disponibilitatea componentei şi 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:

$$Disponibilitatea\_Componentei = \sum_{j=1}^{m} Disponibilitate\_F\_Componenta\ (C_k, S, T_P) / m$$

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 acelaşi timp pentru a suporta şi a furniza acelaşi set de caracteristici ca o „cutie neagră".

Se consideră C_{DE}= {C_0, C_1, …, 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:

$$Disponibilitate\_Extinsa\_Componenta = (C_{DE}, S, T_P) = Disp\_Ext - timp(C_{DE}) / [Disp\_Ext - timp(C_{DE}) + Indisp\_Ext - timp(C_{DE})],$$

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 $T_{1\,(DE)}$, $T_{2\,(DE)}$, … şi $T_{m\,(DE)}$ 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} este activă şi disponibilă pentru a pune la dispoziție toate funcțiile componentei. Similar se calculează şi *Indisp_Ext – timp*(C_{DE}):

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

unde $T'_{1\,(DE)}$, $T'_{2\,(DE)}$, … şi $T'_{m\,(DE)}$ sunt intervalele de indisponibilitate din T_P.

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

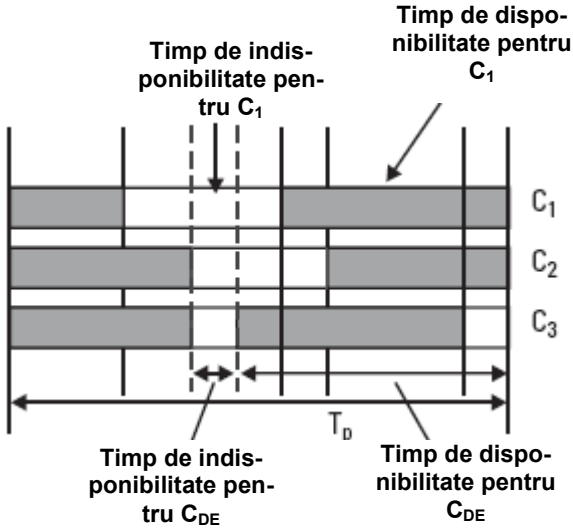


Fig.3. Componentă cu disponibilitate extinsă (C_{DE}) cu trei componente şi timpii de disponibilitate şi indisponibilitate

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 componentei şi timpul total de evaluare $T_{p'}$, care include atât timpii de funcționare, cât şi timpii de nefuncționare.

Plecând de la această definiție, siguranța în funcționarea componentelor poate fi uşor evaluată dacă se aplică o componentă de urmărire şi înregistrarea a timpilor de funcționare şi 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:

$$Sigur\_Comp_{Tp}(C_k) = timp\_func(C_k) / |T_p|$$

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

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:

$Sigur\_Comp_{Tp}(C_{DE}) = timp\_func(C_{DE}) / timp\_func(C_{DE}) + timp\_nfunc(C_{DE}) / |T_p|$,

unde, $timp\_func(C_{DE})$ este timpul de funcționare a componentei C_{DE}, iar $timp\_nfunc(C_{DE})$ cumulează timpii de nefuncționare şi reparare a componentei.

Pentru a măsura $timp\_nfunc(C_{DE})$, se va presupune că $C_{DE} = \{C_{HA1}, \ldots, C_{HAn}\}$ 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 $T_{P'}$ şi 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 componentei C_{DE} bazată pe disponibilitatea pentru serviciile funcționale. Prin această abordare, o defecțiune a serviciilor componentei C_{DE} pe timpul unui interval de timp presupune că toate componentele redundante ale C_{DE} au eşuat în furnizarea serviciilor lor. Astfel, se utilizează o cale diferită de a calcula timpul de funcționare al componentei C_{DE}. Pentru o componentă de tip N-cluster $C_{DE} = \{C_{HA1}, \ldots, C_{HAn}\}$, $timp\_func(C_{DE})$ se va calcula astfel:

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

unde T_j este un interval de timp din $T_{P'}$ şi cel puțin una din componentele C_{DE} este funcțională şi poate furniza servicii funcționale în timpul T_j.

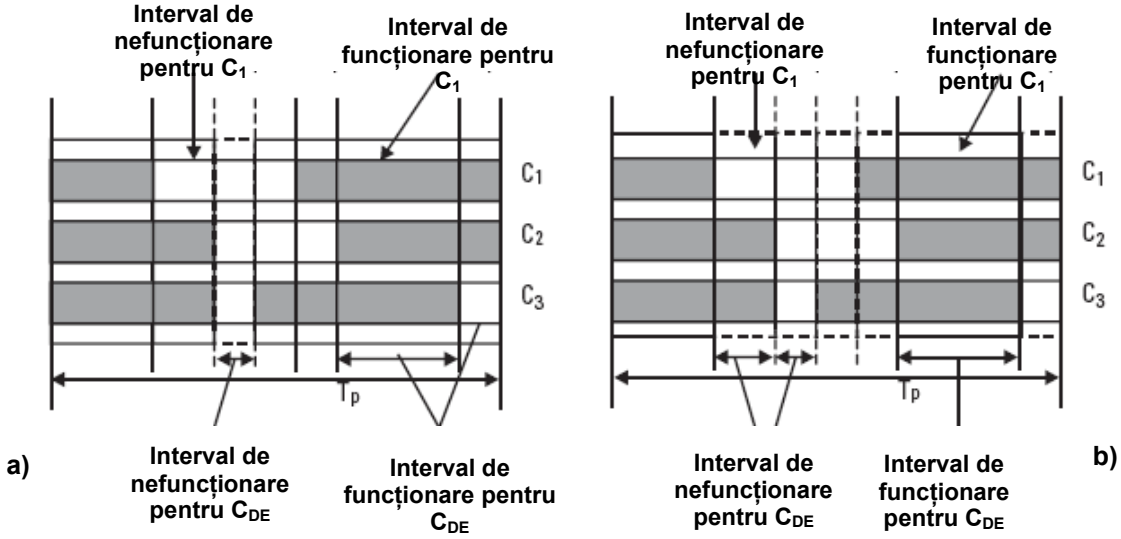


Fig.4. Comparația între cele două criterii privind siguranța în funcționare

Metrici privind rezultatele

Acest tip de metrici este conceput pentru a măsura rata de succes a unei componente software în procesarea evenimentelor, mesajelor şi 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 tranzacţii procesate cu succes.

Pe parcursul perioadei de testare şi evaluare, se pot aplica o serie de teste pentru determinarea maximului, minimului şi mediei pentru rezultatele tranzacţiilor la fiecare tip de tranzacţie.

Bazându-ne pe metrica de rezultat se poate defini rata rezultatului tranzacţiei a unei componente C_k ca fiind raportul dintre numărul total de cereri de tranzacţii efectuate cu succes şi numărul total de cereri de tranzacţii primite pe perioada de evaluare $T_{p'}$:

$$rata\_rez\ (C_k, T_p, TR) = m / n$$

unde *m* este numărul total al cererilor de tranzacţii TR procesate cu succes în timpul $T_{p'}$, iar *n* este numărul total al cererilor de tranzacţii TR primite în aceeaşi 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 tranzacţii de categoria TR.

Metrici de scalabilitate

Un sistem sau componentă se consideră scalabilă dacă poate fi dezvoltată astfel încât să furnizeze eficient funcţii şi servicii într-un domeniul predefinit de şabloane. Validarea scalabilităţii şi evaluarea unui sistem şi a componentelor sale se face pentru studiul:

- limitelor componentelor sau sistemului în procesare prin prisma dimensiunii problemelor, volumului de date şi a numărului de utilizatori cu acces concurent;
- creşterii performanţelor şi degradărilor în cazul diferitelor configuraţii şi setări.

Funcţia marginală a unei componente

Pentru a măsura limitele performanţei unei componente (C_k) trebuie observate cantitativ modificările performanţei în execuţie pentru o funcţie de dimensiune fixă în cazul creşterii firelor de execuţie concurente din C_k .

Aplicând teorema Amdahl privind performanţa componentelor software şi anume că *viteza întregului sistem este influenţată de cea mai lentă componentă*, se va defini timpul de execuţie al unei funcţii F_J din cadrul unei componente C_k astfel:

$$T(Ck, F_J, n) = Tserial + Tconc / n$$

unde timpul de execuţie $T(Ck, F_J, n)$ este împărţit în două: timpul consumat de elementele seriale din C_k, notat cu T_{serial} şi timpul de execuţie al *n* diferite fire de execuţie concurente, notat cu T_{conc}. Pe măsură ce numărul de fire de execuţie concurente creşte, cu atât valoarea timpului total de execuţie de apropie de valoarea T_{serial}. În practică, timpul total de execuţie tinde către valoarea maximă (numit *punct de prag*) pe măsură ce sunt apar fire de execuţie concurente în apelurile componentei C_k. Apariţia de noi fire concurente, va solicita şi mai mult sistemul prin suprasolicitare, ceea ce va influenţa şi timpul de execuţie al componentelor seriale, în sensul creşterii.

Se observă că această metrică poate fi folosită în măsurarea acceleraţiei pentru timpul de execuţie al funcţiei F_j când numărul firelor de execuţie concurente al componentei C_k creşte de la *n* la *m*:

$$T_{accel}{}^{Fj}(n,m) = T(Ck, F_J, n) - T(Ck, F_J, m)$$

Valorile negative ale acceleraţiei, desigur, indică o încetinire a execuţiei, valorile pozitive reprezentând o execuţie mai rapidă.

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

Limitele rezultatelor unei tranzacţii şi îmbunătăţirea clusterelor componentei

Pentru a creşte capacitatea unui sistem de aplicaţie, de obicei se utilizează componente de timp N-cluster (care reprezintă o grupare de componente identice) care suportă un set specific de funcţii. Astfel evaluarea modificărilor privind rezultatele unei procesări pentru o tranzacţie se face prin determinarea plajelor de valori între care se situează nivelul rezultatelor şi 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 tranzacţii 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 tranzacţii 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 componentei:
$Rezultat_{sistem\_imb}(n, m) = Rezultat_{sistem}(T_{set}, m, TR, T_p) - Rezultat_{sistem}(T_{set}, n, TR, T_p)$
unde $Rezultat_{sistem\_imb}(n, m)$ reprezintă îmbunătățirea rezultatelor atunci când numărul componentelor identice de tip cluster creşte 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;