מבוא לבדיקות עומסים | נחום דימר

נחום דימר

הינו יזם ומנכ"ל **TEST4LOAD**, חברה מובילה בתחום בדיקות עומסים ושיפור ביצועים של מערכות תוכנה. הוא מעורב בפיתוח, בדיקות וניהול של מגוון פרויקטים בתחום תוכנה ומערכות ארגוניות. יש לו ניסיון עשיר ורקע טכנולוגי מעמיק במגוון תחומים הכוללים הנדסת תוכנה וביצועים, וכן אבטחת איכות של מערכות אינטרנט וטלקום.
TEST4LOAD עוזר לתכנן, ליישם, לבדוק ולמטב את ביצועים של מערכות תוכנה מרובות משתמשים. בין לקוחות החברה נמנים חברות וגופים מובילים בארץ ובעולם. http://cloudbeat.io/

חלק ב'

בחלק א' דיברנו על עקרונות ותכנון של בדיקות עומסים. בחלק הזה נדבר על הצדדים המעשיים יותר – באילו כלים להשתמש, איך לבנות ולהריץ את הבדיקה ואיך לנתח את התוצאות.

בחירת הכלי

נתחיל מבחירת כלי לביצוע הבדיקה. מתוך מבחר הכלים הרחב בשוק, רובם הם מסחריים אך ישנם גם מספר כלים חינמיים. ישנם ארבעה הבדלים עיקריים בין הכלים שכדאי לשים לב אליהם

- שפת ושיטת פיתוח של התסריטים
- תמיכה בפרוטוקולים שונים
- יכולת ניתוח התוצאות והדו"חות
- יכולת להריץ כמויות גדולות מאוד של משתמשים (עשרות אלפי משתמשים)

ברשימה להלן אפרט את הכלים הפופולריים ביותר ובאיזה מקרים שימוש באיזה כלים יהיה יעיל ביותר:

JMeter

JMeter הוא כלי חינמי מבוסס קוד פתוח שמאפשר לבצע בדיקות עומסים בעיקר למערכות ווב אך גם תומך במספר פרוטוקולים נוספים. זהו כלי פשוט מאוד לשימוש שאינו דורש שום ידע תכנותי. הוא מתבסס על Java ולכן ניתן להשתמש בו בכל מערכות הפעלה, כולל Linux ו-Mac. תאורטית, אין מגבלה לעומס שניתן לחולל בעזרת הכלי. אך בבדיקות של אלפי משתמשים, כאשר יש צורך במספר מכונות העמסה, העבודה עם הכלי יכולה להיות מאתגרת ולא טריוויאלית. כלי מאוד מומלץ לבדיקות של מערכות ווב, במיוחד בשילוב עם כלי נוסף בשם בלייזמטר.

Blazemeter

בלייזמטר היא פלטפורמה להרצת בדיקות JMeter בענן. בלייזמטר נותנת מענה לשני החסרונות העיקריים של JMeter: חוסר מנגנון דו"חות וקושי בהרצת בדיקות גדולות – והיא עושה זאת בצורה מעולה. שימוש בבלייזמטר מומלץ מאוד במקרים של בדיקת יישומי ווב, ממשקי תוכנה (API) ואתרים מותאמים למובייל, במיוחד כשמדובר בכמויות גדולות של משתמשים (מעל 1,000 משתמשים). בלייזמטר גם מאפשרת להתחבר בקלות לכלי CI כמו Jenkins ו-TeamCity וכלי ניטור כמו NewRelic.

WebLOAD

וובלואד הוא אחד מהכלים הוותיקים בשוק של חברת RadView הישראלית והוא מבוסס JavaScript, מה שנותן גמישות גדולה בפיתוח התסריטים אך עדיין נשאר פשוט לשימוש ולא מצריך ידע טכני או תכנותי מעמיק. למרות שהכלי מכוון בעיקר למערכות ווב הוא תומך גם במספר פרוטוקולים נוספים. הוא שילוב מנצח של המחיר ויכולות מתקדמות כגון תמיכה רחבה יותר בפרוטוקולים השונים והפקת דו"חות. מומלץ לארגונים שמחפשים כלי מסחרי במחיר סביר עם תמיכה טובה ויכולות ניהוליות.

LoadRunner

LoadRunner של חברת HP נחשב ל-Rolls-Royce של בדיקות העומסים. הוא אחד הכלים הוותיקים והטובים בשוק והוא תומך במספר רב של פרוטוקולים. בנוסף לטכנולוגיות ווב, הוא מאפשר לבדוק טכנולוגיות של Flex, Java, Oracle, Citrix, SAP ועוד רבים אחרים. הכלי מתחבר למערכות נוספות של HP כגון Quality Center ו-BAC. החיסרון העיקרי של הכלי הוא המחיר. לכן, מומלץ להשתמש בו אם בודקים בטכנולוגיות שכלים אחרים לא תומכים בהן (דוגמת Citrix ו-SAP). אם מדובר ביישומי ווב, כדאי לחפש חלופה זולה יותר.

הקלטת התסריט

חשוב להדגיש שבדיקות עומסים מתבצעות בדרך כלל ברמת הפרוטוקול ולא ברמת ה-UI. כלומר, בדיקות עומסים מדמות תעבורה ולא פעולות של משתמש על המסך. זאת מכיוון שמטרת הבדיקה היא ליצור עומס על צד השרת, שהוא משאב משותף לכל המשתמשים וביצועיו תלויים במישרין בכמות המשתמשים (זהו העומס). כך פעולת משתמש שלא יוצרת פניה לשרת, לא מעניינת אותנו במסגרת בדיקת עומסים.

לעיתים, עבודה ברמת תעבורה מורכבת יותר מאשר עבודה עם ממשק משתמש, אך החדשות הטובות הן שכל כלי בדיקות העומסים תומכים בהקלטה של התסריט – מה שמאפשר לבנות את התסריט הראשוני במהירות וללא צורך בהתעמקות בפרטי התעבורה שנוצרת בין צד השרת לצד הלקוח. אך הקלטה בלבד אינה מספיקה בדרך כלל, ויש צורך בהתאמת התסריט, במיוחד בטיפול במשתנים דינמיים, כפי שיוסבר בהמשך.

ההקלטה מתבצעת לרוב דרך מנגנון של Proxy. כלי הבדיקה מתחזה ל-Web Proxy ע"י שינוי הגדרות הדפדפן ומעביר את כל התעבורה דרכו. כל בקשת HTTP מוקלטת ונשמרת בפורמט או בשפה שנתמכת ע"י הכלי. חלק מהכלים (לדוגמה- LoadRunner) מקליטים את התעבורה של האפליקציה שנבדקת בלבד וחלק מהכלים יקליטו את כל התעבורה שהמחשב המקומי מייצר (כמו JMeter). כדאי לסגור את כל התוכנות והדפדפנים ולהשאיר רק את המערכת הנבדקת פתוחה.

טיפול במשתנים דינמיים (Correlation)

למרות שכל הכלים מאפשרים לבצע הקלטה של התסריט, ברוב המקרים הרצה של התסריט המוקלט לא תעבוד ויתקבלו שגיאות HTTP שונות ומשונות. זה קורה בגלל שכמעט בכל אפליקציית רשת מודרנית ישנם משתנים דינמיים. מטרת משתנים אלו היא להעביר מידע בין בקשות

מבוא לבדיקות עומסים | נחום דימר

HTTP עוקבות. פרוטוקול HTTP, בהגדרתו, הוא חסר מצב (stateless) כלומר הבקשה לא מכילה מידע לגבי בקשות קודמות ואם יש צורך בניהול מצב, על השרת לעשות זאת בעצמו בעזרת משתנים שמחוללים בצד השרת ונשלחים עם כל בקשת המשך. דוגמה טובה למשתנה דינמי הוא sessionId בחלק מהאתרים הכתובים ב-Java או VIEWSTATE באתרים שנכתבו ב-NET. ערך של משתנה דינמי נקבע בדרך כלל בתחילת הסשן – חיבור ראשוני של המשתמש למערכת. יש צורך להוציא את הערך הראשוני מדף מסויים ולהעבירו הלאה לכל שאר הבקשות שעושות בו שימוש. פעולה זו נקראת קורלציה (correlation). קורלציות הן חלק מרכזי בהכנה של התסריט לריצה ומצריכות בדרך כלל הכי הרבה זמן בשלב פיתוח התסריטים. רוב הכלים המסחריים מגיעים עם יכולת לאיתור וטיפול (החלפה) של המשתנים הדינמיים, אך אין פתרון שעובד ב-100% מהמקרים. לכן אני ממליץ להשקיע זמן ולהבין את הפרוטוקול בו המוצר הנבדק עובד. כמו כן, מומלץ ללמוד להשתמש בביטויים רגולריים (Regular Expressions).

טרנזקציות

אם נקליט ברצף את כל הפעולות שבתסריט הבדיקה, ייווצר תסריט ארוך עם מאות בקשות HTTP. בזמן ההרצה יהיה מאוד קשה להבין איזו בקשה שייכת לאיזו פעולה. גם מנהלים וגם מפתחים רוצים קודם כל לדעת באיזו פעולה או דף יש בעיה ורק אחר כך לרדת לרמת הבקשה הבודדת (דף אחד או פעולה יכולים לייצר עשרות ומאות בקשות HTTP). לכן, יש צורך לחלק את כל בקשות ה- HTTP לפעולות או לדפים רלוונטיים. טרנזקציה (transaction) מאפשרת לקבץ בקשות HTTP לפעולות עסקיות ולמדוד אותן יחד. טרנזקציה הוא רעיון נפוץ בכל כלי בדיקות העומסים ובעזרתו ניתן לבודד ולהתמקד באזורים מסויימים במערכת כאשר מתגלה בעיה.

חשוב מאוד לא להקליט את כל התסריט ברצף אלא לחלק אותו כבר בזמן ההקלטה לטרנזקציות. בחלק מהכלים, בעיקר באלו המסחריים, ניתן לעשות זאת בקלות ע"י שימוש בסרגל כלים שמופיע על המסך בזמן ההקלטה. ב-Jmeter יש צורך לחזור לממשק המשתמש ולהחליף את הענף שבו נקלטות הבקשות (לא הכי נוח אך בהחלט סביר).

טיפול בתמונות, קובצי JS, CSS וכו'

חלק גדול מהתעבורה שתיווצר כתוצאה מההקלטה יהיו קבצי תמונות, JavaScript, CSS וכו'. קבצים אלו הם חלק ממשאבי הדף ונקראים Resources. כל הכלים יודעים לטפל בהם בצורה אוטומטית ולרוב אין צורך להשאיר אותם בסקריפט. חשוב לעבור על כל הבקשות ולמחוק את אלו עם הסיומות הרלוונטיות. LoadRunner, למשל, מאגד את הקבצים הנוספים באופן אוטומטי. אם הבקשות האלו יישארו בתסריט זמן התגובה של הטרנזקציה עלול להיות כפול.

זמני המתנה

כאשר משתמש אמיתי עובד על האתר או המערכת, הוא מבצע פעולות באיטיות וישנו מרווח זמן משמעותי בין הפעולות. לפעמים עוברות עשרות שניות בין לחיצה על הכפתור וקריאה של המסך החדש לבין הקלדה או פעולה נוספת כלשהי. אם רק נקליט את התעבורה ולא נכניס זמני המתנה שרירותיים, כלי העומסים ידמה פעילות "רובוטית" – הדמיה מהירה של הבקשות ללא המתנה כלל. זה יצור עומס לא מציאותי על השרת ולעיתים "יחנוק" את השרת סתם, ללא כל בסיס אמיתי לכך. פעולות כמו הבנת המסך או קריאת טקסט יוצרות זמני המתנה ארוכים וצריך לשמר התנהגות דומה גם בעת הבדיקה.

זמן ההמתנה בין הפעולות חשוב מאוד בבדיקות עומסים וחשוב לשמר אותו קרוב למציאות ככל שניתן. רוב הכלים יודעים להקליט את זמן ההמתנה בין הפעולות באופן אוטומטי. זמן זה נקרא think time או sleep time בז'רגון המקצועי. כדי לתת אחידות לפעולות השונות ולנרמל את התוצאות, אני ממליץ לחלק את זמני ההמתנה לשתי קטגוריות: זמן המתנה קצר וזמן המתנה ארוך, וזאת בהתאם לפעולות של המשתמש. לדוגמה: הכנסת שם משתמש וסיסמה ולחיצה על כפתור כניסה הן פעולות מהירות וזמן ההמתנה בהן יהיה קצר. לעומת זאת, קריאת טקסט ארוך או מעבר על הדוח הן פעולות ארוכות . באופן שרירותי, אני בדרך כלל קובע זמן המתנה קצר ל-10 שניות וזמן המתנה ארוך ל-30 שניות, אך זה כמובן יכול להיות שונה בין מערכת למערכת.

> **כמות בקשות בשנייה הוא המדד היחיד שמייצג את העומס האפקטיבי שנוצר בזמן הריצה על המערכת**

מה מודדים?

הרבה חושבים שכלים לבדיקות עומסים מביאים פתרון קסם: מגדירים תסריטי בדיקה והכלי אוטומטית מגלה בעיות בביצועים ומראה לנו איפה הן נמצאות. לצערנו, לא כך הדבר (לפחות בינתיים).
כדי להפיק תועלת מרבית מהבדיקה, צריך קודם כל להבין אילו מדדים קיימים, מה משמעותם ומה הקשר ביניהם.
להלן רשימה של מדדים עיקריים שיש כמעט בכל הכלים (שם המדד יכול להשתנות מכלי לכלי, אך משמעותו נשארת זהה):

- **Connect Time** – הזמן שלוקח לצד הלקוח (דפדפן) ליצור חיבור לשרת בטרם מועברים עליו נתונים כלשהם. אם הערך הזה גבוה, משמעותו שיש בעיה עם קו התקשורת או ששרתי וב עמוסים כל כך שאין באפשרותם לפתוח חיבורים חדשים. לעיתים מסתתרים במדד זה מרכיבים נוספים כגון זמן חיפושDNS ופתיחת חיבור SSL.
- **Latency / Time to First Byte** – זמן השהייה מרגע שליחת הבקשה (לאחר שהחיבור נוצר) ועד לקבלת הבייט הראשון של התשובה. זהו מדד חשוב מאוד כי הוא מייצג את זמן עיבוד הבקשה בצד השרת. אם הערך של המדד הזה גבוה, משמעותו שהאפליקציה מגיבה לאט באחת מהשכבות שלה (בד"כ front-end, middleware או database), מה שילווה לרוב בצריכה מוגברת של משאבי השרת (כגון מעבד, זיכרון או דיסק) באחת השכבות של המערכת (וזו גם דרך לזהות את האזור הבעייתי).
- **Response Time** – זמן תגובה כולל של כל הבקשה, מרגע השליחה ועד לקבלת התשובה במלואה (בחלק מהכלים, response time יכלול גם את latency ואף connect time). זהו מדד שכיח ביותר שמשתמשים בו לדיווח זמן תגובה של המערכת. latecy נמוך ו-response time גבוה מצביעים בדרך כלל על מגבלה של רוחב פס או בעיות רשת שונות. כאשר ערכם של שני המדדים גבוהים, מדובר בדרך כלל בבעיות אפליקטיביות או בעיות בבסיס הנתונים.

מבוא לבדיקות עומסים | נחום דימר

- **Transaction Time** – זמן תגובה של טרנזקציה ספציפית – מטרת המדד היא לקבץ ביחד את זמני התגובה של בקשות בודדות המייצגות תהליך עסקי מסויים. מדד זה משלים את מדד response time ומאפשר ראיה עסקית על תוצאות הבדיקה. הרבה פעמים, משתמש עסקי פנים ארגוני לא מכיר את הבקשות הבודדות שמרכיבות את המערכת, ומדד זה מאפשר לו להבין משמעויות עסקיות בלי להיכנס לפרטים הטכניים. כמו כן, שימוש בטרנזקציות מקל משמעותית על תהליך ניתוח התוצאות ומאפשר לכל בעלי העניין לדבר בשפה אחת.
- **Requests per Second** – כמות בקשות בשנייה שנשלחות למערכת וזו גם כמות הבקשות שהמערכת מסוגלת לתמוך באותו רגע. למעשה, כשכלי העומסים שולח בקשה למערכת, הוא מחכה לקבל תשובה לפני שהוא ממשיך לבקשה הבאה. אם התשובה מתעכבת, הבקשה הבאה תישלח באיחור אחרי קבלת התשובה לבקשה הקודמת. כלומר, כמות הבקשות בשנייה שהכלי מפיק תלויה בכמות הבקשות שהשרת מסוגל לעבד. זהו מדד העומס האמיתי על המערכת. למרות שבתחילת הבדיקה אנחנו מגדירים את כמות המשתמשים שירוצו בו זמנית, כמות בקשות בשנייה הוא המדד היחיד שמייצג את העומס האפקטיבי שנוצר בזמן הריצה על המערכת. המגמה בגרף של מדד זה אמורה להיות זהה למגמת הגרף של מדד כמות המשתמשים. כלומר, ככל שכמות המשתמשים עולה, גם כמות הבקשות בשנייה אמורה לעלות. אם כמות המשתמשים נשארת קבועה, אז גם המגמה של כמות הבקשות בשנייה אמורה להיות קבועה (גרף של כמות הבקשות בשנייה תמיד תנודתי, לכן חשובה המגמה). אם רואים ירידה חדשה בערך המדד בלי שכמות המשתמשים פוחתת, זהו סימן ראשון לרוויה של המערכת ויצירה של צוואר בקבוק.
- **Transactions per Second** (TPS) – דומה ל-requests per second רק מתייחס לטרנזקציות. פחות משתמשים במדד זה לצורך ניתוח מעמיק אלא יותר לצורך בחינה של עמידה ביעדים עסקיים של הבדיקה. נפוץ להגדיר יעדי הבדיקה על בסיס TPS פר פעולה עסקית.
- **Throughput** (Mbps) – נפח התעבורה שהבדיקה מייצרת, בדרך כלל מחושב על בסיס גודל התשובה שחוזרת. יחידת החישוב משתנה מכלי לכלי. חשוב לשים לב אם מדובר ב-bits או bytes וכדאי להמיר את יחידת החישוב ל-Mbps.. ממדד זה אפשר ללמוד אילו קווי תקשורת פנימיים וחיצוניים (בעיקר חיבור של החווה לרשת האינטרנט) נדרשים כדי לתמוך בצורה תקינה בכמות המשתמשים שהוגדרה בבדיקה ומה צפויה להיות התעבורה היומית במערכת (הנתון הזה יכול להיות חשוב לצורך חישוב עלויות פריסה בענן).

כל הכלים מציגים את המדדים הללו בצורת ערכי מינימום, ממוצע ומקסימום. רוב הכלים גם מאפשרים לראות את ערכי החציון, אחוזון 90 וסטיית תקן. חשוב גם להסתכל על ערכים כמו אחוזון 90 , סטיית תקן וחציון כי ערכים אלו נותנים מבט טוב יותר של חווית המשתמש ומגמת המדד.

> **חשוב גם להסתכל על ערכים כמו אחוזון 95, סטיית תקן וחציון כי ערכים אלו נותנים מבט טוב יותר של חווית המשתמש ומגמת המדד**

ניטור התשתית

כלי לבדיקות עומסים יכול לתת אינדיקציה לבעיה בתשתית – אך מהמדדים של הכלי בלבד לעיתים קשה להבין את מהות הבעיה. כדי להבין ממה נובעת הבעיה, חשוב לנטר ולמדוד גם את השרתים ואף לעיתים את ציוד התקשורת (במיוחד Load Balancer) של המערכת. מבט משולב של צד הלקוח וצד השרת מאפשר להבין במהירות ממה נובעות בעיות האיטיות, השגיאות או חוסר היציבות.
ניתן לנטר את השרתים באופן עצמאי או לבקש מצוות התשתיות לעשות זאת עבורך. אם הניטור מתבצע ע"י גורם חיצוני, כדאי לבקש את הנתונים בקובץ אקסל כדי לשלב את הנתונים עם המדדים שמגיעים מהכלי לבדיקת העומסים. חשוב להתחיל לנטר את השרתים זמן קצר לפני תחילת הבדיקה, במהלך הבדיקה ופרק זמן של כ-10 דקות אחרי סיום הבדיקה. כך יהיה ניתן להבין האם המערכת הייתה עמוסה לפני תחילת הבדיקות ואיך המערכת "מתקררת" בסיום הבדיקה.

ישנם מגוון רחב של כלים לניטור הקלסי של השרתים ולרוב הארגונים יש כבר כלי מועדף. אך רוב הכלים הקיימים כיום בארגונים מבצעים ניטור של משאבי השרת ולא נכנסים לרמה האפליקטיבית. ניטור רגיל של משאבי השרת עוזרים להבין את האזור או השכבה בה יש בעיה אך לא מאפשרים לשים אצבע על מקום מסויים, כגון רכיב, שורת קוד או שאילתת SQL. לכן, אמליץ כאן על כלי לניטור אפליקטיבי שיצא לי לעבוד איתו בשנים האחרונות ושעזר לי לגלות בעיות מורכבות במהירות וביעילות שיא.

הכלי הזה נקרא **NewRelic** שבטח רבים מכם כבר מכירים. זוהי פלטפורמה שמאפשרת לא רק לנטר את משאבי השרת, אלא גם לנטר ביצועים ברמת הקוד ושאילתות בבסיס הנתונים. בעזרתו ניתן בקלות להבין באיזה שרת, באיזו קומפוננטה ואף באיזו שורת קוד או שאילתת SQL נמצאת הבעיה. NewRelic קל מאוד להתקנה והפעלה ומקצר משמעותית את תהליך הבדיקה ותיקון התקלות. מחירו לא גבוה במיוחד ולכן מומלץ בחום.

ניטור חווית המשתמש

רוב הכלים של בדיקות העומסים עובדים ברמת הפרוטוקול ולא לוקחים בחשבון את זמן הבניה של הדף בצד הדפדפן. כמו כן, מודל ההתקשרות מול השרת לא תמיד זהה לזו של הדפדפנים הנפוצים. לכן לעיתים נוצרים פערים בזמני תגובה בין מה שהכלי מדווח לבין מה שמשתמש חווה במציאות. ברוב המקרים, הפערים הללו לא גדולים וניתן להתעלם מהם, אך במערכות בהם יש הרבה אינטגרציה של רכיבי צד ג' או הרבה מרכיבים שונים (resources), הפערים הללו יכולים להיות משמעותיים.

במקרה של רשת ארגונית שפרוסה בכל העולם או אתר שהלקוחות מגיעים אליו מאזורים שונים בעולם, חשוב להבין את חווית המשתמש מאזור גאוגרפי מסויים.

מבחינה מעשית, ניתן לנטר את חווית המשתמש בכמה דרכים: הדרך הפשוטה ביותר היא ניטור ידני שנעשה בעזרת שימוש ב-Developer Tools של הדפדפן המועדף. בכל אחד מהדפדפנים ניתן למצוא חלון המאפשר למדוד זמן ואת אופן טעינת הדף, כולל כל הבקשות המרכיבות אותו (בד"כ נמצא תחת מקש Network). יש לשים לב איזה בקשות לוקחות יותר משנייה ומה השפעתן על זמן הטעינה של כל הדף.

דרך נוספת היא שימוש באחד מהכלים המאפשר לנטר את חווית המשתמש על בסיס אוטומציה של ממשק משתמש. רובם מבוססים על Selenium. כלי שאני ממליץ עליו הוא אתר פשוט וחינמי בשם WebPageSpeed.

האתר מאפשר להכניס כתובת של הדף ולקבל פירוט של זמן טעינת הדף מכל מקום בעולם. מתאים לתסריטים פשוטים ולא ארוכים.

ישנם מספר כלים שמאפשרים לבנות תסריטים מורכבים וארוכים יותר שלא מצריכים הכנסה ידנית של כתובת הדף ויודעים לרוץ בתדירות מסויימת. לבדיקות מורכבות וארוכות, סוג זה של כלים נוח יותר ולא מצריך הפעלה ידנית בזמן הבדיקה. אחד מהכלים מסוג זה הוא מוצר של NewRelic, שכבר הזכרנו לעיל. זהו מוצר נפרד של NewRelic שנקרא **Synthetic Monitoring** ומצריך תשלום נוסף. אם אתם משתמשים כבר במוצרים של NewRelic אז כדאי להשתמש גם בו. אפשרות נוספת כחול לבן היא שירות של חברת CloudBeat (גילוי נאות: אני שותף בחברה) שמאפשר הרצה של תסריטי חווית משתמש מבוססי Selenium גם מישראל וגם מחו"ל.