**שקף 3**

אז מה זה profiler ?

המטרה העיקרית של profiler היא למדוד את כל –CPU ticks שהתוכנה שלנו מבלה על ה-execution של האפליקציה שלנו.

מה הם cpu ticks ?

Cpu ticks הם יחידת מדידה של Internal system time . יש בדרך כלל counter של מערכת ההפעלה עבור Ticks . ה-current time and date שמשתמשים בהם הפונקציות השונות של מערכת ההפעלה נגזרות מה-counter הזה .

כמה milliseconds מייצג tick זה תלוי במערכת ההפעלה, וזה יכול להשתנות בין התקנות . מנגנון מערכת ההפעלה עושה convert מ-ticks ל-seconds .

יש גם memory profiler , שבהם משתמשים כדי לגלות memory leaks , אבל אנחנו נדבר רק על performance profilers כאן.

**שקף 4:**

אז ל-Node.js יש profiler שהוא built in , וניתן להריץ אותו בעזרת פקודה פשוטה מה-command line .

הוא מגיע עם כל התקנה של Node.js . ה-profiler אוסף את כל המידע לתוך log file וזהו .

לא ניתן לקרוא את הלוג הזה. כלומר שאתה צריך שיהיה לך tool שיהפוך את ה-log file ומשהו קריא לעין אנושית. מגיע built in עם node.js גם tool שיכול להציג את ה-output של ה-profiler בצורה שאפשר לקרוא אותה ולהבין מה הבעיות. הסתכלות על הקובץ עצמו לא תיתן לכם כלום.

**שקף 5:**

ל-Node.js יש flag שעושה enable ל-profiler , ה-flag הזה הוא –prof . הוא בעצם אוסף CPU ticks לתוך log file בתוך אותה ספריה היכן שממוקם ה-main script של האפליקציה שלך.

אתם צריכים להריץ את הפקודה node.js –prof server.js כדי להריץ את האפליקציה עם הפרופיילר.

עכשיו, יש לכם log file עם מידע על ה ticks . שם ה-log file יהיה isolate-\*.log . כדי לנתח את ה-log משתמשים בפקודה

node --prof-process isolate-0x101804a00-v8.log

**שקף 6:**

בואו נדמיין שהדוגמה הבאה היא קטע קוד מפרוייקט גדול שמייצר לנו בעיית ביצועים. בחרתי את המודול crypto בגרסה הסינכרונית שלו, וספציפית את pbkdf2 מכיוון שהוא פוגע מאוד בביצועים לעומת הגרסה הא-סינכרונית , ויהיה לנו מאוד קל לאתר את הבעיה ב-log .

**שקף 8:**

אז בואו נריץ את התוכנה עם ה-built in tool ש-node מספק לנו, נפתח את הקובץ isolate..log ונראה מה קורה .

אז עכשיו יש לנו מידע על ה-Ticks שהוא human readable ואנחנו יכטלים להבין מה קורה.

אז בקובץ הlog יש לנו כמה Sections :

Shared libraries

JavaScript

C++

Summary

C++ entry Points

Heavy Profile

ב-heavy profile נקבל את המידע לגביי מה ה-heavy functions . ואנחנו רואים ש-100% מה-ticks הלכו על הפונקציה הסינכרונית של הקריפטו. לא מדוהר פה על 100% אלא קרוב ל-100%, וזהו צוואר הבקבוק שלנו. אחרי שאנחנו יודעים את זנ אפשר להחליף את crypto לגרסה הא-סינכרונית שלו.

**שקף 9:**

אני רק רוצה לחזור על ה-sections שיש בניתוח ה-log .

**שקף 10:**

מה שמאוד חשוב זה לראות שה-heavy profile עשתה רק קצת יותר מ-40 ticks לעומת יותר מ2000 בגרסה הסינכרונית, וזה כבר מראה למו על שיפור מאוד גדול בביצועים שלנו, כאשר הפונקציות הכבדות עשו בסך הכל עד 40 ticks בסה"כ.