

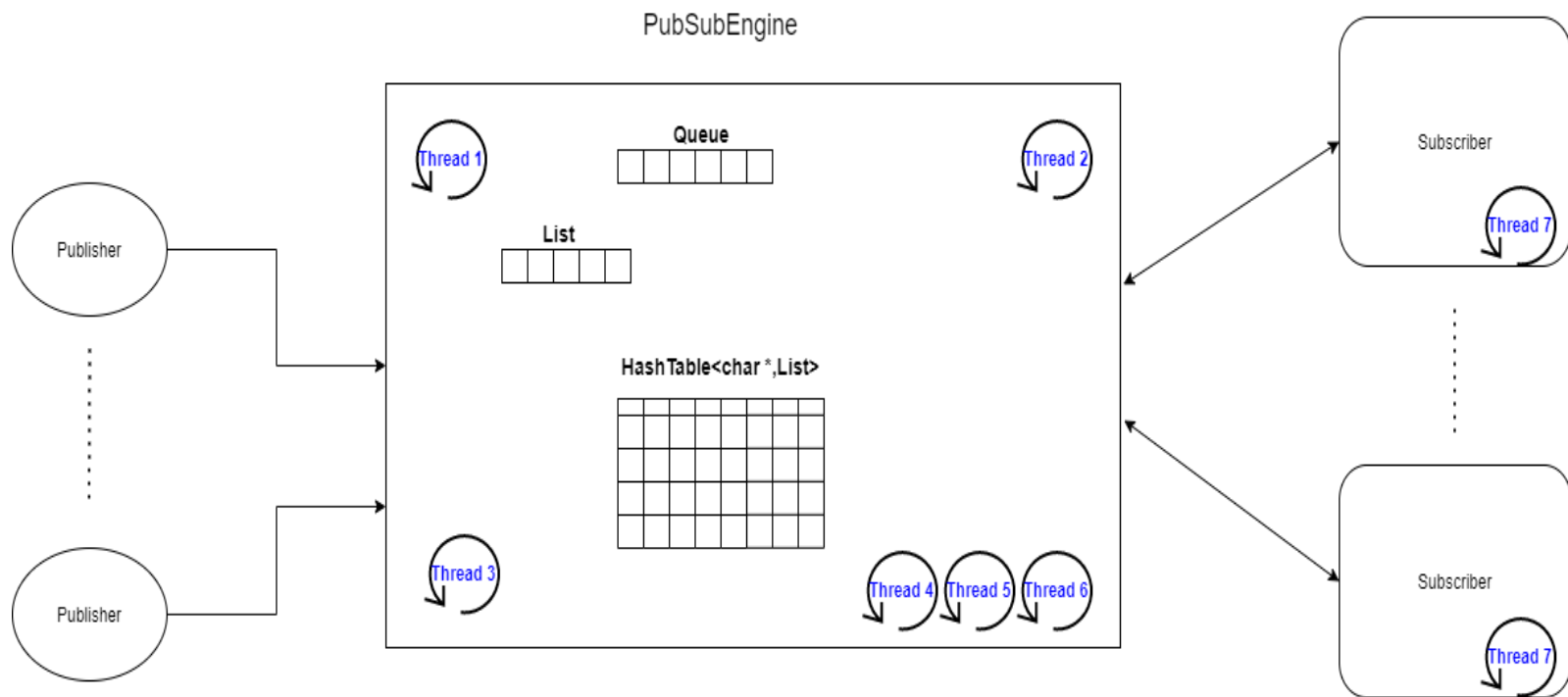
Publisher-Subscriber

1. Uvod

Potrebno je napraviti PubSub servis čiji je zadatak da obezbedi distribuiranje artikala između komponenti Publisher i Subscriber.

Cilj projekta je da omogući optimalnu razmenu podataka između proizvoljnog broja komponenti sistema.

2. Dizajn



-*Publisher* → Konzolna aplikacija koja se konektuje, kreira artikle i šalje ih na PubSubEngine.

-*Subscriber* → Konzolna aplikacija koja se konektuje na PubSubEngine i vrši prijavljivanje na topic za koji želi da dobija vesti. Komponenta sadrži Thread 7 koji omogućava prijavu na više topica pritiskom na taster 'x'.

-*PubSubEngine* → Konzolna aplikacija koja predstavlja centralnu tačku sistema. Komponenta sadrži sledeće niti:

- Thread 1 – prihvata konekcije publisher-a i smešta njihove socket-e u listu
- Thread 2 – prihvata konekcije subscriber-a i smešta njihove socket-e u listu, takođe proverava događaje nad prihvaćenim socket-ima subscriber-a i u slučaju pretplate na novi topic dodaje socket subscriber-a u HashTabelu, a u slučaju prekida konekcije određenog subscriber-a uklanja ga iz Liste i HashTabele
- Thread 3 – proverava događaje nad prihvaćenim socket-ima publisher-a i u slučaju prijema novog artikla smešta taj artikal na Queue, a u slučaju prekida konekcije određenog publisher-a, uklanja ga iz Liste konektovanih
- Thread 4,5,6 – preuzimaju poruke sa Queue i na osnovu topica pronalaze u HashTabeli prijavljene subscriber-e i prosleđuju im artikal.

Thread 4,5,6 predstavljaju ThreadPool kao optimalno rešenje problema distribucije artikala subscriber-ima.

3. Strukture podataka

Od struktura podataka u projektu su implementirani: Queue, List, HashTable.

U nastavku slede opisi i zadaci struktura.

-*Queue* → FIFO struktura čiji je zadatak privremeno čuvanje pristiglih podataka strukture tipa Artikal, koja sadrži topic i text. Operacije Enqueue i Dequeue omogućavaju dodavanje i skidanje sa queue, respektivno.

-*List* → Dva primerka ove strukture služe za čuvanje konektovanih publisher-a i subscriber-a, pomoću strukture tipa Uticnica koja u sebi sadrži njihov socket. Ova struktura podataka je korišćena zbog mogućnosti pretrage, dodavanja i brisanja uz dinamičku alokaciju memorije.

-*HashTable* → Predstavlja kolekciju koja sadrži strukturu tipa Subscribers koja se sastoji od topic-a i List-e prijavljenih subscriber-a na taj topic. Ova struktura je izabrana jer je optimalno rešenje za dobavljanje List-e subscriber-a prijavljenih na traženi topic.

4. Rezultati Testiranja

Prvi stres test – Publisher šalje 1000 kratkih poruka za isti topic. Proverava se opterećenje mreže.

SummaryEventsMemory UsageCPU Usage

Take Snapshot

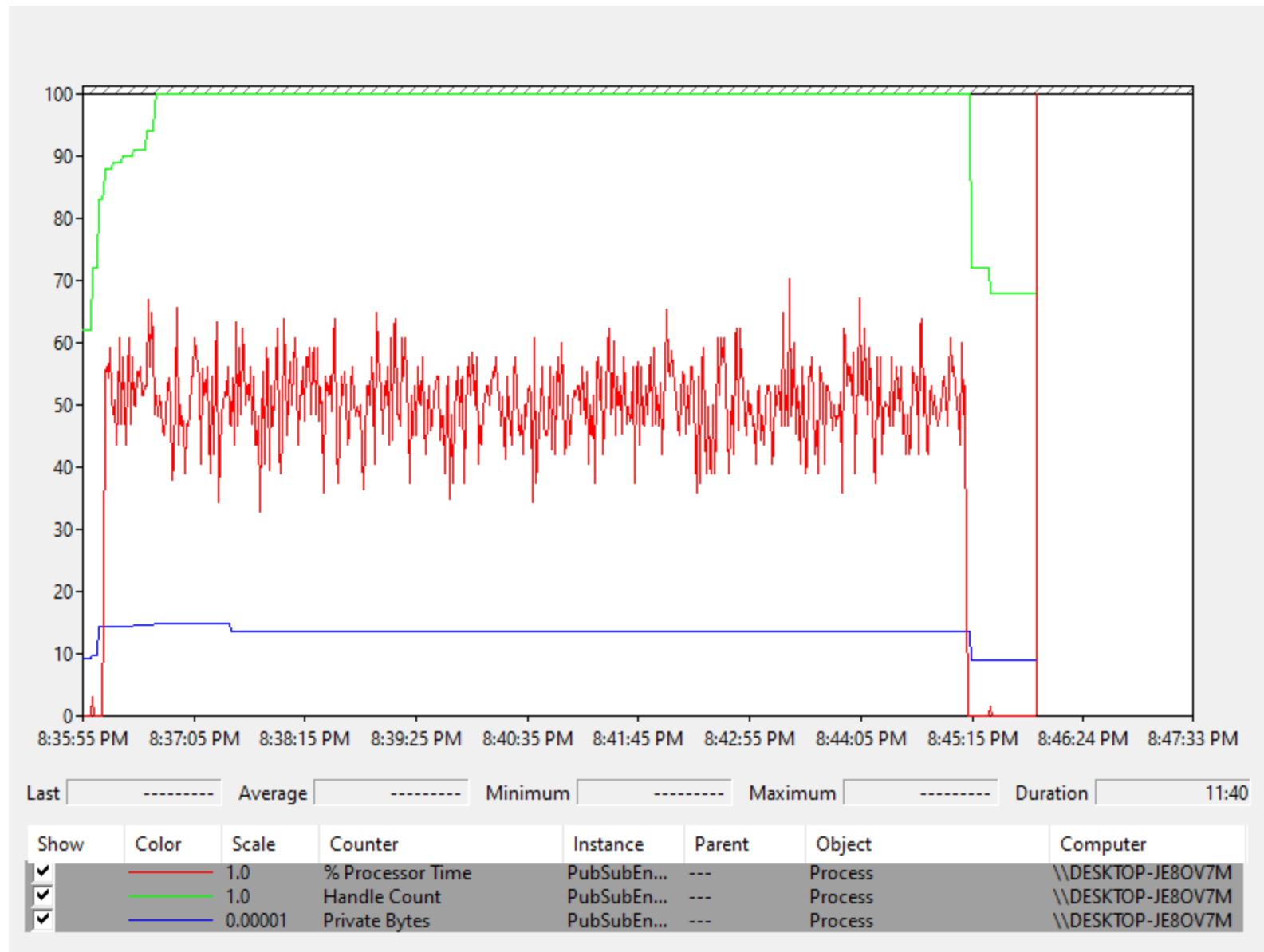
View Heap

Heap Profiling

Delete

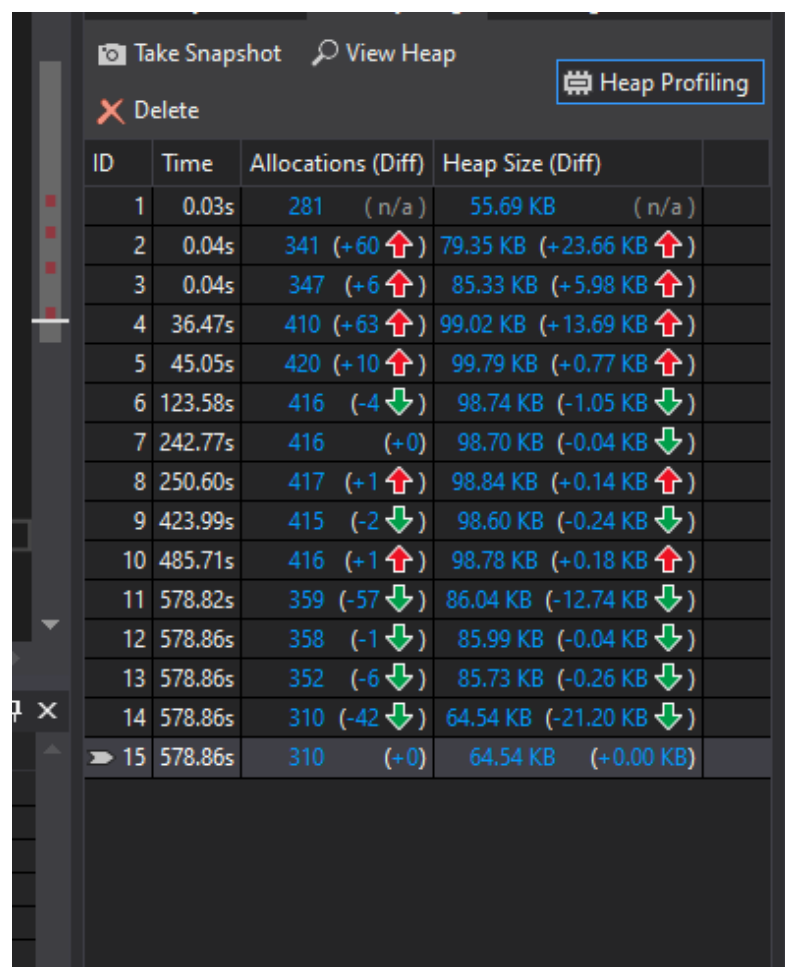
ID	Time	Allocations (Diff)	Heap Size (Diff)	
1	0.03s	281 (n/a)	55.69 KB (n/a)	
2	0.06s	341 (+60 ↑)	79.35 KB (+23.66 KB ↑)	
3	0.06s	350 (+9 ↑)	87.20 KB (+7.86 KB ↑)	
4	23.57s	385 (+35 ↑)	96.47 KB (+9.27 KB ↑)	
5	28.69s	393 (+8 ↑)	97.27 KB (+0.79 KB ↑)	
6	43.53s	399 (+6 ↑)	97.73 KB (+0.46 KB ↑)	
7	181.98s	394 (-5 ↓)	96.50 KB (-1.23 KB ↓)	
8	309.53s	395 (+1 ↑)	96.64 KB (+0.14 KB ↑)	
9	453.69s	395 (+0)	96.64 KB (+0.00 KB)	
11	542.81s	342 (-53 ↓)	82.45 KB (-14.19 KB ↓)	
12	542.86s	341 (-1 ↓)	82.41 KB (-0.04 KB ↓)	
13	542.86s	338 (-3 ↓)	82.28 KB (-0.13 KB ↓)	
14	542.86s	296 (-42 ↓)	61.08 KB (-21.20 KB ↓)	
➤ 15	542.86s	296 (+0)	61.08 KB (+0.00 KB)	

Slika 1. Izgled Heap-a prvog stres testa



Slika 2. Preformance monitor grafik prvog stres testa

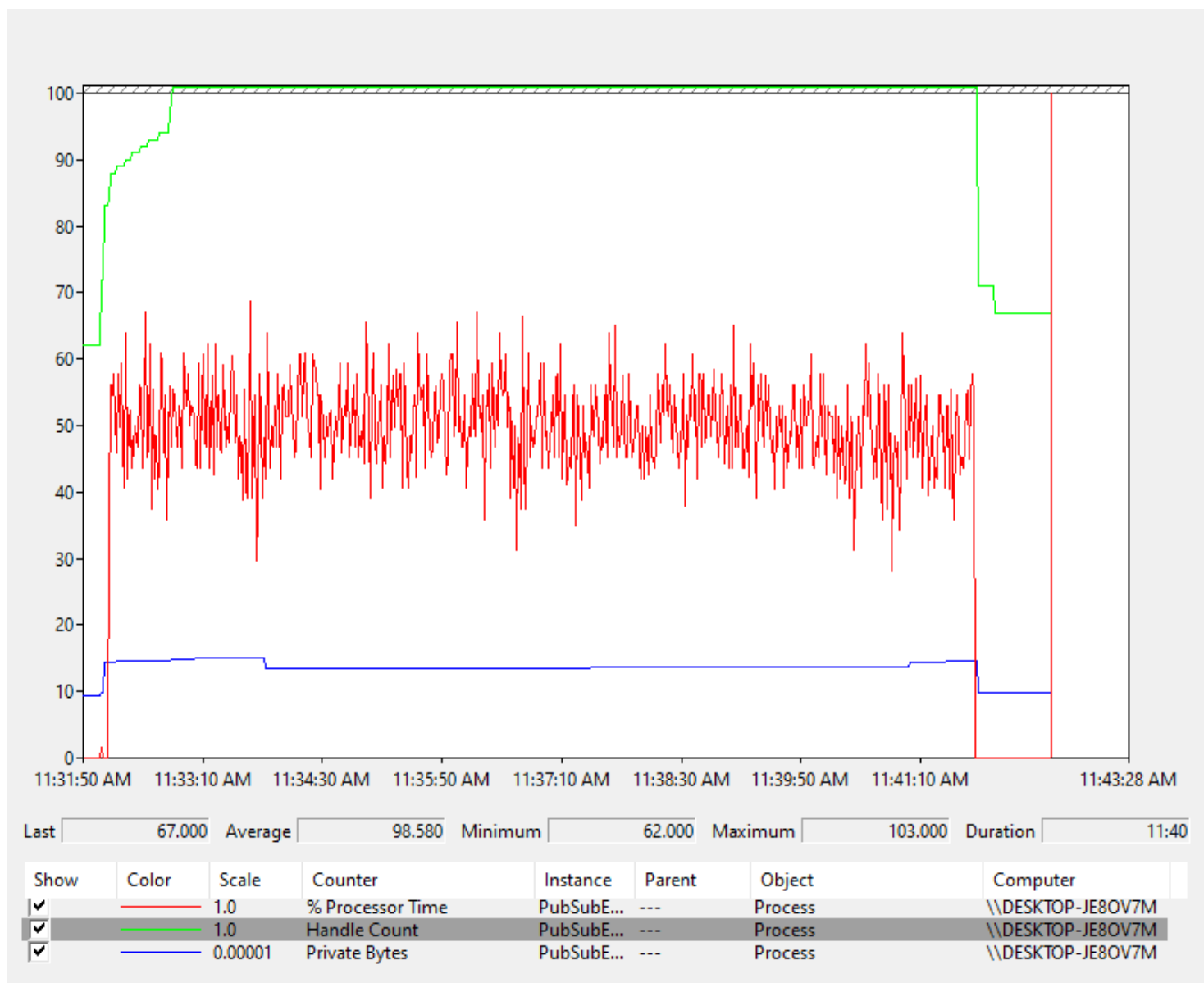
Drugi stres test – Publihser šalje 1000 dužih poruka naizmenično na dva različita topic-a. Ovim testom se takođe proverava opterećenost mreže i brzina isporuke subscriber-ima.



The screenshot shows the 'Heap Profiling' window of a development tool. At the top, there are buttons for 'Take Snapshot', 'View Heap', and 'Heap Profiling' (which is highlighted with a blue box). Below these is a 'Delete' button with a red 'X' icon. The main part of the window is a table with the following columns: 'ID', 'Time', 'Allocations (Diff)', and 'Heap Size (Diff)'. The table contains 15 rows of data, showing a sequence of heap allocations and deallocations over time. The 'Allocations (Diff)' column uses red upward arrows for increases and green downward arrows for decreases. The 'Heap Size (Diff)' column uses the same color coding. The table is currently sorted by 'Time' in ascending order.

ID	Time	Allocations (Diff)	Heap Size (Diff)
1	0.03s	281 (n/a)	55.69 KB (n/a)
2	0.04s	341 (+60 ↑)	79.35 KB (+23.66 KB ↑)
3	0.04s	347 (+6 ↑)	85.33 KB (+5.98 KB ↑)
4	36.47s	410 (+63 ↑)	99.02 KB (+13.69 KB ↑)
5	45.05s	420 (+10 ↑)	99.79 KB (+0.77 KB ↑)
6	123.58s	416 (-4 ↓)	98.74 KB (-1.05 KB ↓)
7	242.77s	416 (+0)	98.70 KB (-0.04 KB ↓)
8	250.60s	417 (+1 ↑)	98.84 KB (+0.14 KB ↑)
9	423.99s	415 (-2 ↓)	98.60 KB (-0.24 KB ↓)
10	485.71s	416 (+1 ↑)	98.78 KB (+0.18 KB ↑)
11	578.82s	359 (-57 ↓)	86.04 KB (-12.74 KB ↓)
12	578.86s	358 (-1 ↓)	85.99 KB (-0.04 KB ↓)
13	578.86s	352 (-6 ↓)	85.73 KB (-0.26 KB ↓)
14	578.86s	310 (-42 ↓)	64.54 KB (-21.20 KB ↓)
15	578.86s	310 (+0)	64.54 KB (+0.00 KB)

Slika 3. Izgled Heap-a za drugi stres test



Slika 4. Performance monitor grafik drugog stres testa

5. Zaključak

Prvi stres test – na osnovu izgleda Heap-a na slici 1 i grafika na slici 2 može se zaključiti da ne dolazi do curenja memorije. Vrednosti su unutar prihvatljivih granica. Na grafiku (slika 2) zelena linija koja pokazuje broj HANDLE-ova se po završetku testa vraća na približno istu poziciju početnoj, razlika u visini se objašnjava postojanjem main niti koja završava sa radom tek po gašenju aplikacije. Opterećenje procesora (crvena linija) je ravnomerno.

Drugi stres test – rezultati testa (slika 3 i slika 4) nam govore da u drugom testu kao i prvom ne dolazi do curenja memorije, vrednosti su takođe unutar dozvoljenih granica. Zelena linija (broj HANDLE-ova) se u ovom slučaju ponaša kao i u prvom testu. Opeterećenje procesora je očekivano.

6. Potencijalna unapređenja

Sistem se može unaprediti dodavanjem više od tri niti u pool, ali za naše potrebe tri niti su sasvim dovoljne. Više niti omogućava efikasniju isporuku.