**Realizacija prikaza zauzetosti resursa Android TV aplikacije na STB uredjaju**

*Sara Piljak 2020/0185*

***Elektrotehnički fakultet u Beogradu***

Smer: računarska tehnika i informatika

Mentor: Dejan Babić

Datum:

Sadržaj

[**Uvod** 3](#_Toc174105611)

[**Teorijski okvir** 4](#_Toc174105612)

[1. Osnove Android Operativnog Sistema i Aplikacija 4](#_Toc174105613)

[2. Upravljanje Memorijom u Android Sistemima 4](#_Toc174105614)

[3. Upravljanje Procesorom u Android Sistemima 4](#_Toc174105615)

[4. Zaključak 5](#_Toc174105616)

[**Metodologija** 6](#_Toc174105617)

[1. Merenje Zauzetosti CPU-a 6](#_Toc174105618)

[2. Merenje Korišćenja Memorije 6](#_Toc174105619)

[**Analiza Zauzetosti Resursa** 8](#_Toc174105620)

[1. Merenje CPU i Memorije 8](#_Toc174105621)

[2. Simulacija Opterećenja 8](#_Toc174105622)

[3. Analiza Korišćenja Resursa 8](#_Toc174105623)

[4. Rezultati i Diskusija 9](#_Toc174105624)

[5. Zaključak 9](#_Toc174105625)

## **Uvod**

U današnjem digitalnom dobu, efikasno upravljanje resursima postalo je ključno za razvoj visokokvalitetnih i pouzdanih aplikacija. Ovaj rad istražuje temu „Realizacija prikaza zauzetosti resursa Android TV aplikacije na STB (Set-Top Box) uređaju“, sa posebnim fokusom na analizu i prikaz zauzetosti memorije i procesora u kontekstu Android TV aplikacija.

Android TV platforma, kao specijalizovana verzija Android operativnog sistema, nudi jedinstvene mogućnosti i izazove u oblasti aplikacijskog razvoja. Set-Top Box (STB) uređaji koji koriste Android TV predstavljaju specifičan ekosistem u kojem se aplikacije često suočavaju sa ograničenim resursima, što može uticati na performanse i korisničko iskustvo. Razumevanje i upravljanje resursima kao što su CPU i memorija je stoga od suštinskog značaja za razvoj aplikacija koje nude glatko i stabilno iskustvo.

U ovom radu, fokusiraćemo se na tri ključne aspekte:

1. **Analiza Zauzetosti Memorije i Procesora**: Razmotrićemo kako se memorija i CPU resursi koriste unutar Android TV aplikacije. Ova analiza će uključivati praćenje i merenje potrošnje resursa, identifikaciju potencijalnih problema kao što su curenje memorije ili neefikasno korišćenje CPU-a, i implementaciju tehnika za optimizaciju.
2. **Prikaz Zauzetosti Resursa na Ekrenu**: Razvijaćemo metodologiju za prikaz statistike zauzetosti resursa na ekranu. Ovo uključuje kreiranje vizualnog interfejsa koji korisnicima omogućava da u realnom vremenu prate performanse aplikacije. Fokusiraćemo se na dizajn korisničkog interfejsa i implementaciju funkcionalnosti za prikaz podataka na ekranu Android TV aplikacije.
3. **Korišćenje Tehnologija**: Korišćenje nativnih programskih jezika kao što su C i C++ u kombinaciji sa Java i Kotlin jezicima pruža moćne alate za precizno upravljanje resursima i optimizaciju performansi. Ovaj rad će koristiti Java Native Interface (JNI) za integraciju nativnog koda i demonstrirati kako se ove tehnologije mogu koristiti za efikasno prikupljanje i prikaz podataka o resursima.

Razumevanje kako različiti delovi aplikacije utiču na resurse i kako se ovi resursi mogu prikazati korisnicima u realnom vremenu je ključno za razvoj aplikacija koje ne samo da funkcionišu optimalno već i nude korisnicima uvid u performanse aplikacije. Kroz detaljnu analizu i implementaciju prikaza zauzetosti resursa, ovaj rad će pružiti značajne uvide u poboljšanje performansi Android TV aplikacija na STB uređajima.

Cilj ovog istraživanja je da se pruže praktične preporuke za razvoj aplikacija koje efikasno koriste resurse i pružaju korisnicima kvalitetno iskustvo gledanja. Kroz sistematski pristup analizi, implementaciji i optimizaciji, ovaj rad će doprineti boljem razumevanju i unapređenju tehnologija koje pokreću moderni Android TV ekosistem.

## **Teorijski okvir**

### 1. Osnove Android Operativnog Sistema i Aplikacija

Android je otvoreni operativni sistem razvijen od strane Google-a, koji je prvenstveno dizajniran za mobilne uređaje poput pametnih telefona i tableta. Od svog lansiranja, Android je evoluirao kako bi se prilagodio različitim vrstama uređaja, uključujući Android TV i STB (Set-Top Box) uređaje. Android operativni sistem baziran je na Linux kernelu i pruža platformu za razvoj aplikacija koje mogu pristupiti i koristiti resurse uređaja na način koji je u velikoj meri upravljiv i kontrolisan.

Android aplikacije se razvijaju koristeći Java i Kotlin programske jezike, uz podršku za nativne biblioteke napisane u C i C++ preko JNI (Java Native Interface). Aplikacije koriste različite komponente kao što su aktivnosti, servisi, emitatori, i sadržajni provajderi, da bi funkcionisale unutar Android okruženja.

### 2. Upravljanje Memorijom u Android Sistemima

Memorija u Android sistemima igra ključnu ulogu u obezbeđivanju glatkog rada aplikacija i operativnog sistema. Android koristi sofisticirane tehnike za upravljanje memorijom koje uključuju:

1. **Zoniranje Memorije**: Android koristi različite nivoe memorije, uključujući RAM i virtualnu memoriju. Uređaji sa većim RAM-om mogu obraditi više aplikacija u isto vreme, dok virtualna memorija omogućava aplikacijama da koriste više prostora nego što je fizički dostupno u RAM-u.
2. **Prioriteti Procesa**: Android koristi različite prioritete za procese i aplikacije. Aktivne aplikacije i procesi imaju viši prioritet u odnosu na pozadinske procese, što omogućava bolju raspodelu memorije i resursa prema potrebama korisnika i sistema.
3. **Memorijska Ograničenja**: Android platforma postavlja ograničenja na količinu memorije koju aplikacije mogu koristiti, čime se smanjuje rizik od prekomernog korišćenja memorije i stabilnosti sistema.

### 3. Upravljanje Procesorom u Android Sistemima

Procesor je ključna komponenta u svakom računarskom sistemu, uključujući Android uređaje. Upravljačka jedinica procesora obavlja sve operacije vezane za izvršenje instrukcija i operacije nad podacima. U kontekstu Android sistema, upravljanje procesorom obuhvata:

1. **Multitasking i Višenitne Operacije**: Android podržava multitasking i višenitno izvođenje, što omogućava aplikacijama da obavljaju više zadataka u isto vreme. Upravljački sistem koristi višezadatnost (multitasking) i višenitne (multithreading) pristupe kako bi optimizovao rad procesora i omogućio istovremeno izvršavanje više aplikacija i operacija.
2. **Prvenstveno Korišćenje Procesora**: Aplikacije u Android sistemima mogu biti dizajnirane tako da koriste procesor na efikasan način, minimizirajući opterećenje procesora kada nije potrebno. Ovo uključuje optimizaciju algoritama, upravljanje nitima, i minimizaciju intenzivnih izračunavanja.
3. **Praćenje i Optimizacija Korišćenja Procesora**: Android pruža alate i API-je za praćenje i optimizaciju korišćenja procesora. To uključuje aplikacije za praćenje performansi koje prate korišćenje CPU-a i omogućavaju razvojnom timu da identifikuje i reši probleme sa performansama.
4. **Zauzetost Procesora**: Zauzetost procesora u Android sistemima može se pratiti korišćenjem različitih alata i tehnika. Ovo uključuje analizu koliko je CPU opterećen i koliko resursa koristi aplikacija. Takođe, pravilno upravljanje procesima i nitima može pomoći u smanjenju nepotrebne potrošnje CPU-a i poboljšanju opšte efikasnosti sistema.

### 4. Zaključak

Razumevanje upravljanja memorijom i procesorom u Android sistemima je od suštinskog značaja za razvoj aplikacija koje su efikasne i pouzdane. Efikasno upravljanje memorijskim i procesorskim resursima doprinosi boljem korisničkom iskustvu, dugotrajnosti baterije, i stabilnosti sistema. Kroz primenu tehnika kao što su automatsko upravljanje memorijom, multitasking, i praćenje resursa, razvijači mogu optimizovati performanse svojih aplikacija i osigurati da one funkcionišu optimalno na različitim uređajima.

## **Metodologija**

Metode za Merenje Zauzetosti CPU-a i Memorije

U cilju analize performansi aplikacije, važno je precizno merenje zauzetosti CPU-a i memorije. U ovom radu, implementirane su različite metode za prikupljanje i analizu tih podataka, koristeći funkcije koje omogućavaju praćenje resursa u realnom vremenu. U nastavku su opisane primene i funkcionalnosti svake od metoda.

### 1. Merenje Zauzetosti CPU-a

Za merenje zauzetosti CPU-a, korišćene su funkcije koje preuzimaju podatke iz sistema i izračunavaju procenat korišćenja CPU-a. Ključne funkcije uključuju:

* **read\_cpu\_stats**: Ova funkcija čita podatke o CPU vremenu iz /proc/stat datoteke. Ovi podaci uključuju vreme korisničkog procesa, vreme sistema, vreme u stanju "nice" i vreme kada je CPU bio u stanju "idle". Funkcija koristi ove informacije za praćenje ukupnog CPU vremena i izračunavanje procenta zauzetosti.
* **prikaziZauzetostCpu**: Ova funkcija koristi read\_cpu\_stats za prikupljanje statistike o CPU-u u dva vremenska trenutka. Razlika između ovih vremenskih trenutaka se koristi za izračunavanje procenta korišćenja CPU-a. Funkcija izračunava i prikazuje trenutni procenat opterećenja CPU-a u odnosu na prethodni period.
* **get\_cpu\_usage\_for\_pid** i **get\_cpu\_usage\_for\_pid\_2**: Ove funkcije su specifične za praćenje CPU vremena za određeni PID. get\_cpu\_usage\_for\_pid koristi podatke iz /proc/[PID]/stat, dok get\_cpu\_usage\_for\_pid\_2 koristi naprednije merenje koje omogućava preciznije praćenje promene u CPU vremenu. Ove funkcije takođe izračunavaju CPU opterećenje kao procenat, koristeći razliku između prethodnih i trenutnih očitavanja.
* **simulate\_cpu\_load**: Ova funkcija simulira različite nivoe opterećenja CPU-a koristeći zadati faktor opterećenja (load\_factor). Tokom simulacije, funkcija kreira intenzivne CPU operacije i odgovarajuće "idle" periode kako bi se simuliralo različito opterećenje procesora.

### 2. Merenje Korišćenja Memorije

Za merenje korišćenja memorije, korišćene su sledeće funkcije:

* **get\_memory\_usage\_for\_pid**: Ova funkcija čita podatke iz /proc/[PID]/status datoteke kako bi prikupila informacije o zauzetosti memorije za dati PID. Konkretno, funkcija čita vrednost RSS (Resident Set Size) koja predstavlja količinu fizičke memorije koju proces koristi.
* **simulate\_memory\_allocation\_c**: Ova funkcija simulira opterećenje memorije tako što alocira i popunjava memorijske blokove sa karakterima, prema zadatom faktoru opterećenja (load\_factor). Funkcija alocira memoriju u veličini od 1 MB i koristi je tokom simulacije. Nakon završetka, memorija se oslobađa.
* **read\_from\_file**: Ova funkcija čita podatke iz /proc/meminfo datoteke kako bi prikupila informacije o slobodnoj i dostupnoj memoriji. Ovi podaci pomažu u analizi kako alokacija i oslobađanje memorije utiču na ukupnu dostupnost memorije u sistemu.

Sve ove funkcije omogućavaju sveobuhvatno merenje performansi CPU-a i memorije, i omogućavaju analizu kako različite operacije i opterećenja utiču na resurse sistema. Korišćenjem ovih metoda, moguće je prikupiti precizne podatke o resursima i optimizovati rad aplikacije za bolje performanse.

## **Analiza Zauzetosti Resursa**

U ovoj sekciji ćemo razmotriti kako se resursi sistema, poput CPU-a i memorije, koriste i prate u tvom kodu. Ova analiza uključuje razumijevanje kako različite funkcije i niti utiču na korišćenje CPU i memorijskih resursa.

### 1. Merenje CPU i Memorije

Merenje resursa se obavlja kroz nekoliko ključnih funkcija koje koriste /proc fajl sistem na Linuxu. Ove funkcije omogućavaju prikupljanje informacija o trenutnom opterećenju CPU-a i korišćenju memorije.

* **Merenje CPU Zauzetosti**:
  + Funkcija read\_cpu\_stats prikuplja osnovne podatke o CPU opterećenju.
  + Funkcija prikaziZauzetostCpu koristi ove podatke za izračunavanje procenta korišćenja CPU-a. Ova funkcija meri promene u CPU vremenu između dva uzorka i koristi ih za proračun procenta zauzetosti CPU-a.
* **Merenje Korišćenja Memorije**:
  + Funkcija read\_from\_file čita podatke iz /proc/meminfo i omogućava praćenje slobodne i dostupne memorije pre i posle alokacije.
  + Funkcija get\_memory\_usage\_for\_pid prikuplja informacije o fizičkoj memoriji (RSS) za dati PID, što omogućava detaljno praćenje memorijskog zauzeća procesa.

### 2. Simulacija Opterećenja

* **Simulacija CPU Opterećenja**:
  + Funkcija simulate\_cpu\_load kreira intenzivno CPU opterećenje koristeći beskonačne petlje sa malim obradama i prebacuje se između aktivnog i neaktivnog vremena. Ovo omogućava simulaciju različitih nivoa opterećenja CPU-a.
* **Simulacija Korišćenja Memorije**:
  + Funkcija simulate\_memory\_allocation\_c alocira memoriju u velikim blokovima, popunjava ih i kasnije oslobađa. Ova funkcija pomaže u simulaciji uticaja različitih nivoa opterećenja memorije na sistem.

### 3. Analiza Korišćenja Resursa

U analizi koda, pratimo kako se resursi koriste tokom izvršenja niti i simulacija:

* **Kreiranje i Praćenje Niti**:
  + Funkcije worker, worker1, i worker2 kreiraju različita opterećenja CPU-a i memorije u različitim nitima. Svaka nit koristi malloc za alokaciju memorije i zatim simulira CPU opterećenje. Praćenje resursa se vrši pre i posle alokacije i oslobađanja memorije.
* **Praćenje Promena u Resursima**:
  + Prikupljanje i prikazivanje podataka o CPU i memoriji pre i posle rada sa nitima omogućava procenu uticaja tih niti na ukupne resurse sistema. Ove informacije omogućavaju analizu kako različiti nivo opterećenja utiču na performanse sistema.

### 4. Rezultati i Diskusija

Analizom rezultata dobijenih iz funkcija za merenje i simulaciju, može se primetiti kako se promene u CPU i memorijskom opterećenju odražavaju na performanse procesa. Npr., povećanje faktora opterećenja u funkciji simulate\_cpu\_load može pokazati kako sistem reaguje na različite nivoe CPU opterećenja i kako to utiče na ukupnu brzinu obrade.

### 5. Zaključak

U ovoj analizi, kod pokazuje kako se resursi sistema prate i analiziraju u realnom vremenu. Korišćenjem funkcija za merenje i simulaciju, moguće je steći uvid u ponašanje CPU-a i memorije tokom različitih operacija i opterećenja. Ova analiza pomaže u optimizaciji i poboljšanju performansi aplikacija kroz bolje razumevanje resursnih zahteva i opterećenja.