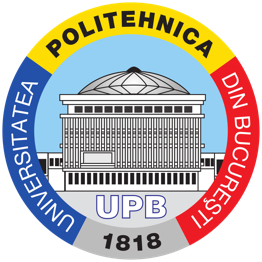
**UNIVERSITATEA POLITEHNICA BUCURESTI**

**FACULTATEA DE AUTOMATICA SI CALCULATOARE**

**DEPARTAMENTUL CALCULATOARE**

****

****

**PROIECT DE DIPLOMĂ**

**SISTEM DE CĂUTARE RAPIDĂ PRIN VOLUME MARI DE DATE**

**IONIȚĂ COSTEL-COSMIN**

**Coordonator Științific:**

Prof. Dr. Ing. Ciprian Dobre

**BUCUREȘTI**

2018

# **CUPRINS**

[**CUPRINS** 2](#_Toc513759365)

[**SINOPSIS** 3](#_Toc513759366)

[**MULȚUMIRI** 4](#_Toc513759367)

[**1.** **INTRODUCERE** 5](#_Toc513759368)

[**1.1** **Context** 5](#_Toc513759369)

[**1.2** **Problema** 5](#_Toc513759370)

[**1.3** **Obiective** 6](#_Toc513759371)

[**1.4** **Soluția propusă** 6](#_Toc513759372)

# **SINOPSIS**

Adobe Audience Manager este unul dintre cele mai importante produse din colecția Adobe Marketing Cloud, cu rolul de a eficientiza campaniile de marketing prin segmentarea claselor de utilizatori în funcție de mai multe criterii și expunerea acestor segmentări către clienți sub forma unei aplicații web.

Datele a milioane de clienți sunt procesate permanent de către o multitudine de sisteme interconectate ce rulează în cloud-ul celor de la Amazon. Peste 500 de instanțe de mașini virtuale conlucrează pentru a procesa o cantitate de date de peste 1 Peta Byte, aflată în continuă creștere.

Există foarte multe servicii cu diverse roluri, de la job-uri de Map Reduce care prelucreză date, până la servici ce extrag date pentru a crea metrici (Analytics). În momentul în care apare o problemă la un anumit serviciu, trebuie inspectat fișierul de log aferent serviciului respectiv, pentru a diagnostica și rezolva problema. Acest proces este foarte consumator de timp, întrucât inginerul trebuie să intre manual pe mașina corectă (căutarea mașinii poate fi o problemă), să găsească fișierul de log și apoi să caute în acest fișier informația legată de eroarea serviciului respectiv.

Pentru a eficientiza acest proces, am stabilit faptul că este necesar un sistem de agregare și căutare rapidă prin aceste informații.

Integrarea sistemelor open source dezvoltate de către organizația Apache (Flume, Hadoop și Solr) s-a dovedit a fi o decizie foarte bună pentru dezvoltarea acestui proiect iar rezultatele au fost de departe peste așteptări în materie de performanțe.

# **MULȚUMIRI**

Mulțumesc în special profesorului meu coordonator Prof. Dr. Ing. Ciprian Dobre pentru suportul acordat pe parcursul dezvoltării acestui proiect, precum și echipei din partea Adobe cu care am colaborat pentru stabilirea arhitecturii și integrarea sistemului cu infrastructura existentă: Manager Dan Tudor, Inginer Software Nicolae Popa, Inginer Software Senior Cosmin Rentea.

# **INTRODUCERE**

## **Context**

Problema căutării unor informații în volume mari de date este probabil una dintre cele mai importante probleme în care eficiența contează extrem de mult. Interconectarea sistemelor existente are ca efect principal transferul unui volum cât mai mare de date, peste distanțe cât mai mari, ajungând la cât mai mulți oameni.

Compania Adobe Systems este a doua cea mai mare companie producătoare de software din lume și este recunoscută pentru produsele sale de editare media (Photoshop, AfterEffects).

Printre cele mai noi produse dezvoltate de Adobe se numără Adobe Audience Manager, un sistem de segmentare a publicului cu scopul eficientizării campaniilor de marketing. Concret, Audience Manager este o aplicație web în care utilizatorul își poate selecta publicul țintă după mai multe criterii și ulterior poate lansa campanii de marketing pe segmentele de public selectate.

## **Problema**

Adobe Audience Manager rulează peste o infrastructură formată din peste

500 de mașini virtuale în cloud (Amazon este provider-ul ales), fiecare mașină având unul sau mai multe roluri asociate. În principiu, peste aceste mașini rulează servicii precum Hadoop, Solr, Spark, cu rolul de a procesa volume mari de date. Un serviciu ce rulează frecvent este cel de Map Reduce care utilizează job-uri specifice pentru a prelucra datele în funcție de necesitățile de business ale companiei.

Există scenarii frecvente în care un job de Map Reduce termină execuția în mod eronat, ca efect al unei excepții în codul job-ului, precum și servicii care își opresc execuția din diverse motive. Motivul pentru care un serviciu se comportă într-un mod nedorit poate fi analizat printr-o singură metodă: inspectarea fișierului de log aferent serviciului respectiv.

Această procedură implică următoarele acțiuni: găsirea mașinii virtuale pe care rulează servicul respectiv (dacă există mai multe mașini pe care rulează serviciul respectiv, trebuie inspectate toate mașinile respective), conectarea pe mașina respectivă, căutarea fișierului de log și inspectarea fișierului respectiv (care poate avea o dimensiune de ordinul sutelor de MB).

Întrucât acest proces implică un consum relativ mare de timp, este necesar un sistem care să monitorizeze informațiile din fișierele de log și care să pună la dispoziția clientului / dezvoltatorului un mecanism foarte ușor de utilizat prin care acesta să înțeleagă cauzele apariției unei probleme și să înceapă procedura de rezolvare a acesteia.

## **Obiective**

Așteptările pe care un client le are de la un utilitar care se ocupă de gestiunea

fișierelor de log, sunt următoarele:

* Capacitatea de a căuta anumite evenimente după mai multe criterii, precum: dată, oră, job id (dacă evenimentul este asociat unui job de Map Reduce), cuvinte din interiorul mesajului aferent evenimentului.
* Capacitatea de a ordona evenimentele rezultate după dată, oră, alfabetic.
* Obținerea unor rezultate într-un timp rezonabil (de nivelul minutelor) peste un volum de date aproximativ 1 TB.
* Capacitatea de a ajusta politica de ștergere a datelor (datele vechi).
* Capacitatea de a genera diverse statistici legate de natura evenimentelor procesate, de exemplu câte evenimente de un anumit tip (cu tag-ul FATAL / ERROR) au fost indexate; câte evenimente de un anumit tip au fost generate de un anumit job / de o anumită mașină.
* Sistemul trebuie să fie capabil să exporte date într-un anumit format, pentru a fi ușor de integrat cu alte servicii / sisteme.
* Modul de interacțiune al utilizatorului cu sistemul trebuie să fie cât mai ușor posibil: o interfață în linia de comandă și o aplicație desktop cu o interfață minimală.

Scopul general proiectului este dat de reducerea timpului de descoperire și

diagnosticare a unei probleme legate de un comportament nedorit al unui serviciu ce rulează într-o infrastructură de dimensiuni mari, lucru care va duce în timp la creșterea timpului de dezvoltare a produsului prin recuperarea timpului pierdut pentru a căuta informația manual.

Pe de altă parte, sistemul are rolul de a construi o bază solidă pentru un sistem dedicat de gestiune a log-urilor, lucru care implică o parte de obținere a datelor în format structurat (deja dezvoltată în cadrul proiectului) și o parte de aplicare a unor algoritmi de învățare automată (funcționalitate ce poate fi implementată în viitor peste infrastructura existentă) care să genereze diverse rezultate legate de distribuția evenimentelor în timp / pe mai multe mașini, detecția anomaliilor, etc. Practic, sistemul vine ca o linie de bază pentru dezvoltarea unui sistem complex adaptat la necesitățile companiei.

## **Soluția propusă**

Soluția propusă constă din 3 componente cu roluri foarte

importante: o primă componentă responsabilă de agregarea / colectarea datelor; o componentă responsabilă de stocarea datelor (fără pierdere de date), deci este necesar un mecanism de replicare construit odată cu mecanismul de stocare; o componentă de indexare a datelor.  
 În urma unui studiu atent al soluțiilor existente pentru aceste 3 componente, am ajuns la următoarea concluzie în ceea ce privește setul de tehnologii ales:

Pentru partea de agregare am ales Apache Flume întrucât acest sistem este creat să ofere un mecanism de transport sigur al datelor, este foarte extensibil (poate fi adaptat foarte ușor la necesitățile utilizatorului), rulează sub formă de agent, fiind posibilă crearea de ierarhii de agenți care să distribuie încărcarea datelor mult mai eficient.

În ceea ce privește stocarea, Apache Hadoop este cea mai bună soluție întrucât oferă un sistem de fișiere distribuit cu un mecanism de replicare foarte eficient și cu posibilitatea de rula job-uri de Map Reduce care să prelucreze datele în timp ce le transportă către destinație.

Pentru partea de indexare distribuită, Apache Solr este de departe cea mai bună soluție, întrucât pe lângă capacitatea de sharding (împărțirea index-ului pe mai multe noduri) și cea de load-balancing, acesta pune la dispoziție unul dintre cele mai eficiente motoare de indexare open source existente, Apache Lucene, capabil să genereze un index cu un volum egal cu aproape o treime din volumul datelor.

Soluția implică unificarea acestor trei sisteme precum și definirea modului de prelucrare a datelor, ceea ce presupune crearea următoarelor subsisteme:

* Un modul de parsare, cu rolul de a prelua datele din locația în care sunt generate; decomprimarea lor (datele sunt stocate în mod comprimat); parsarea efectivă a datelor utilizând expresii regulate; obținerea unor modele (obiecte de tip JSON) și crearea unor fișiere cu un set serializat de astfel de modele care mai apoi sunt trimise către Flume.
* Un modul de monitorizare a fișierelor care sunt stocate in HDFS cu o logică de pornire a job-ului de indexare, în funcție fie de un anumit număr de fișiere care au ajuns in HDFS, fie de trecerea unui anumit timp, fie de obținerea unui semnal de la o coadă de SQS (Simple Queue Service) din Amazon.
* Un script de rulare a job-ului de indexare (este practic un job de Map Reduce).
* O aplicație client care să poată fi utilizată atât din terminal cât și din interfața grafică, prin care utilizatorul să poată realiza operații de căutare prin datele indexate.