Universitatea Politehnica Bucureşti
Facultatea de Electronică, Telecomunicații şi Tehnologia Informației
Master Ingineria Informației şi a Sistemelor de Calcul
Disciplina: Sisteme de operare avansate

# Sisteme de operare multimedia

# IISC_SOA_MM

Masterand: Radu Mitoi

# Cuprins

# Sisteme de operare multimedia

## Introducere

Sistemele de operare multimedia au aparut datorita necesitatii de a reda filme, muzica si videoclipuri la cerere, folosind un sistem de calcul obisnuit. Fisierele audio si video au caracteristici diferite fata de fisierele text pentru care sistemele de operare curente au fost create. Este asadar nevoie de un un nou tip de sistem de operare care sa le poata trata eficient. De asemenea, modul de stocare si redare impune noi cerinte asupra planificatoarelor si a altor parti ale sistemului de operare.

## Fisierele multimedia

In majoritatea sistemelor de fisiere, un document text consta intr-o insiruire secventiala de biti, fara o structura anume de care sistemul de operare sa tina cont. In sistemele multimedia situatia se complica. Fisierele video si audio in schimb sunt mai complexe intrucat sunt inregistrate de diferite dispositive (senzori de imagine, microfoane) si au o structura interna aparte (numar de *cadre pe secunda* diferit, *rata de esantionare* diferita, etc.) . De asemenea, sunt redate de mai multe dispositive simultan: monitoare, sisteme audio.

Fiecare fisier multimedia trebuie sa tina o evidenta de subfisiere care se cuprinde diferite le formate de redare, subtitrari pentru mai multe limbi, traduceri, etc. Pe langa aceasta evidenta trebuie sa se realizeze si o sincronizare in timp real pentru ca imaginea, sunetul si textul sa nu apara decalate.

## Planificarea proceselor multimedia

Sistemele de operare multimedia difera de cele clasice din 3 puncte de vedere:

- Planificarea proceselor
- Sistemul de fisiere
- Planificarea accesului la mediile de stocare

## Planificare proceselor omogene

Cel mai simplu model de server video este acela care poate suporta redarea unui numar fix de filme, toate folosind aceeasi rata de redare, rezolutie, compresie, etc. In acest caz se utilizeaza o planificare simpla dar eficienta. Pentru fiecare film exista un singur proces (sau *thread*) a carui sarcina este sa citeasca filmul de pe un disc, cadru cu cadru si apoi sa transmita acel cadru utilizatorului. Deoarece toate procesele au prioritati egale si au acelasi volum de informatii de prelucrat, planificarea *round-robin* este suficienta pentru o eficienta foarte buna. Singura adaugire necesara algoritmului de planificare este un mecanism de sincronizare pentru ca fiecare proces sa functioneze la o frecventa corecta.

Pentru a obtine aceasta sincronizare se foloseste un ceas *master* care are o frecventa constanta. La fiecare *tick* al ceasului toate procesele sunt rulate secvential, in aceeasi ordine. Cand un proces isi termina sarcina isi suspenda activitatea si elibereaza procesorul pana cand ceasul *master* lanseaza un nou *tick*. Daca numarul proceselor este destul de mic si toate sarcinile acestora pot fi executate intr-un cadru, planificare *round-robin* este suficienta.

## Planificare proceselor in timp-real

Procesele multimedia au in general parametrii diferiti, avand alt model fata de procesele omogene. Acest lucru se datoreaza numarului variabili de utilizatori, compresiei video sau rezolutiei. In consecinta procesele vor rula la frecvente diferite, cu o sarcina mai mare sau mai mica si cu diferite termene pana la care aceste sarcini trebuie efectuate. Rezulta o serie de procese concurente pentru procesor, fiecare cu termenul si sarcina sa. Se presupune ca sistemul de operare cunoaste frecventa fiecarui proces, cat timp ii ia sa finalizeze o sarcina si care este urmatorul termen.

Pentru a exemplifica mediul in care opereaza un planificator multimedia se considera 3 procese A,B si C. Procesul A ruleaza la 30 ms. Fiecare cadru necesita 10 ms pentru a se finaliza. In absenta concurentei, procesul A va functiona in rafale A1, A2, A3.. etc., cu un interval de 30 ms intre procese. Fiecare procesor poate trata un cadru pe care trebuie sa il finalizeze pana la aparitia urmatorului.

Procesul B ruleaza la 25 ms iar procesul C la 20 ms. Timpul necesar pentru finalizarea unui cadru este de 15 ms pentru procesul B si 5 ms pentru C.

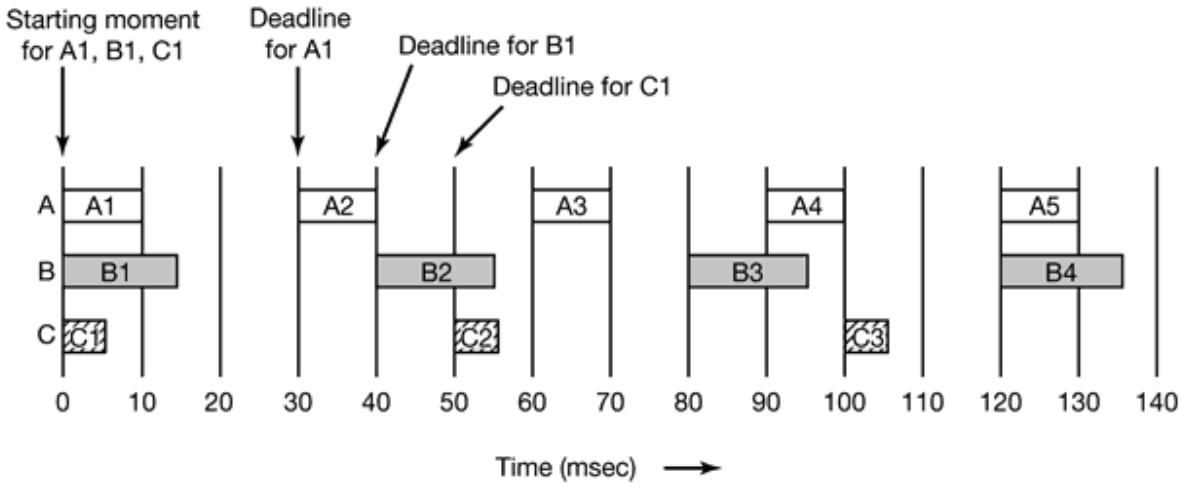


Fig. 1 Exemplu de planificare a proceselor [1]

Se urmareste modul in care se pot planifica A, B si C astfel incat niciunul sa nu-si depaseasca termenul. Mai intai se verifica daca aceste procese sunt planificabile, verificand relatia:

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

, unde Ci este costul procesului (timpul necesar finalizarii unui cadru) iar Pi este perioada procesului . Raportul $\frac{C_i}{P_i}$ indica procentul pe care un proces il foloseste din proceso. Spre exemplu, procesul A consuma 33% din procesor la un moment dat. Daca suma acestor procente este mai mica de 100% atundi procesele sunt planificabile. Calculand se obtine un procent de utilizare al procesorului de 80,8 % deci procesele se pot planifica.

Algoritmii in timp-real pentru planificarea acestor procese pot fi fie statici fie dinamici. Algoritmii statici atribuie fiecarui proces o prioritate fixa dupa care se realizeaza planificarea in functie de aceste prioritati. Algoritmii dinamici nu au o prioritate fixa, aceasta se modifica pe parcursul executiei.

## Planificarea cu rata monotona (RMS)

Planificarea cu rata monotona (*RMS – Rate Monotonic Scheduling)* este varianta clasica de planificare in timp real. Poate fi folosita asupra proceselor care indeplinesc urmatoarele conditii:

1. Fiecare proces periodic se finalizeaza in durata prestabilita.
2. Procesele sunt independente unele de altele.
3. Fiecare proces necesita acelasi procent din procesor pentru fiecare executie.
4. Procesele periodice nu au termen de finalizare.
5. Eliminarea procesului inainte de finalizare se realizeaza instanta, fara nici un surplus de calcul.

*RMS* atribuie fiecarui proces o prioritate fixa egala cu frecventa de aparitie a unui eveniment declansator. Spre exemplu, un proces care ruleaza la fiecare 30 ms ( 33 ori/s) primeste prioritatea 33, un proces care ruleaza la 50 ms primeste prioritatea 20. Prioritatile sunt asadar liniare cu o anumita rata. La rulare, planificatorul lanseaza de fiecare data procesul cu prioritatea cea mai mare, eliminand alte procese daca este cazul. S-a demonstrat ca acest mod de planificare este optim pentru modul static de planificare a proceselor.

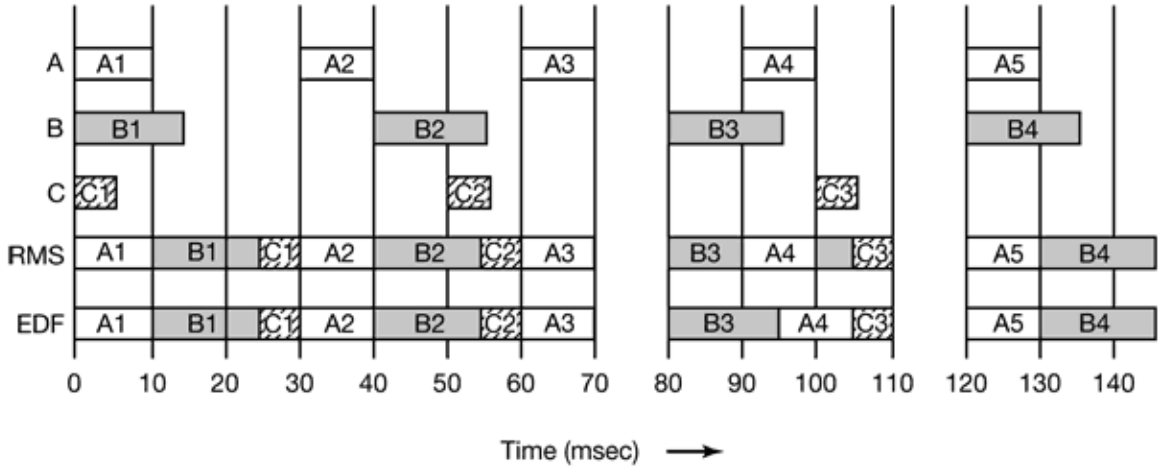


Fig. 2 Exemplu de planificare RMS si EDF [1]

In figura, procesele A, B si C au prioritatile statice 33, 25 si 20. Acest lucru inseamna ca daca procesul A trebuie sa ruleze el va opri executia (daca este cazul) proceselor B si C, procesul B poate opri executia procesului C, iar procesul C trebuie sa astepte eliberarea procesorului pentru a se putea executa. Initial se executa procesul A care se finalizeaza dupa 15 ms, urmat de procesele B si C cu 10 ms respectiv 5 ms. Ciclul se va relua cu procesul A cu prioritate maxima. La t = 90 insa, desi ruleaza procesul B, procesul A trebuie sa ruleze la randul lui si cum are o prioritate mai mare va opri executia procesului B.

## Planificarea Earliest Deadline First (EDF)

Un alt mod de planificare folosit in sistemele de operare in timp real este *Earliest Deadline First*. Acesta este un algoritm dinamic care nu necesita ca procesele sa fie periodice. De fiecare data cand un proces necesita accesul la unitatea de procesare, isi anunta prezenta si durata. Planificatorul mentine o lista a proceselor rulabile, sortate dupa termenul de finalizare. Cand un proces devine disponibil, sistemul verifica daca termenul sau este inaintea procesului curent. Daca da, procesul care ruleaza este oprit si se va lansa noul proces.

Daca se considera 3 procese A, B si C care trebuie sa se finalizeze la momentele 30, 40 respectiv 50, planificarea se desfasoara similar metodei RMS pana la momentul t = 90 cand A trebuie sa porneasca,

dar procesul B ruleaza deja. Desi ambele procese au acelasi termen de finalizare, se va continua executia procesului B deoarece oprirea si repornirea acestuia ar implica un cost suplimentar.

EDF si RMS nu produc rezultate similare in cele mai multe cazuri. In general, algoritmul EDF produce o utilizare mai eficienta a procesorului cu dezavantajul de a fi mai complex si mai dificil de implementat decat RMS. In general, RMS se foloseste atunci can utilizarea procesorului nu este foarte mare. Leyland si Liu au demonstrat ca RMS se utilizeaza eficient daca gradul de utilizare al procesorului este sub ln2 (0.696). Pentru alte cazuri, functionarea acestui algoritm depinde de contextul in proceselor care trebuiesc rulate. Pe de alta parte, EDF functioneaza in orice caz, daca procesele sunt planificabile.

## Sisteme de fisiere multimedia

Sistemele de fisiere multimedia folosesc modele diferite fata de sistemele de operare traditionale. In sistemele de operare obisnuite, pentru a se accesa un fisier, sistemul de operare lanseaza o cerere. Daca aceasta cerere este acceptata, procesul care a lansat-o primeste un *token* pe care il va folosi in cererile ulterioare. Din acest moment procesul poate lansa o cerere de citire, de cate ori va avea nevoie pana cand se va finaliza. Inainte de finalizare va inchide fisierul. Acest model insa nu functioneaza eficient si pentru fisierele multimedia deoarece pentru sistemele in timp-real trebuie sa se cunoasca precis momentele in care fisierul va fi accesat.

### Modelul VCR

Pentru a preintampina aceste neajunsuri se foloseste pentru sistemele de fisiere multimedia un model de tip VCR (*Video Cassette Recorder*). In acest model, pentru a citi un fisier multimedia, un proces lanseaza o cerere specificand ce fisier doreste sa citeasca si cu ce parametrii. Serverul video lanseaza cadre la rata ceruta de proces sarcina tratarii acestora revenind procesului apelant. Daca utilizatorul renunta la vizualizare, fluxul de transmisie este intrerupt. Fluxul se poate intrerupe, se poate sari sau reveni la un anumit moment asemantor sistemelor VCR. Dezavantajul major este ca serverul trebuie sa stocheze aceste informatii in caz ca utilizatorul foloseste functiile de oprire , pentru a se putea continua de la momentul ramas in caz ca utilizatorul doreste sa continue vizualizarea. De asemenea, procesul este complicat si de compresie deoarece cadrele sunt codate independent si trebuie mentinuta o evidenta a ordinii acestora.

### Modelul Near Video on Demand

Transmiterea catre *k* utilizatori a aceluiasi film incarca serverul similar cu transmiterea a *k* filme diferite. Totusi, cu anumite schimbari minor ale modelului se pot face imbunatatiri foarte mari. Problema care apare pentru Video on Demand consta in momentele diferite de timp la care utilizatorii incep vizualizarea. Daca exista un numar mare de utilizatori exista sanse foarte mici ca acestia sa inceapa vizualizarea simultan deci nu pot imparti aceeasi transmisie. Optimizarea se poate realiza facand fiecare utilizator sa astepte anumite momente de timp de la care se va incepe transmisia. Spre exemplu, daca un untilizator doreste sa inceapa vizualizarea la ora 8:02 sistemul de operare va amana momemntul pana la

ora 8:05, fiecare flux incepand din 5 in 5 minute. Daca filmul are durata spre exemplu de 2 ore atunci vor fi necesare doar 24 de fluxuri indiferent cati utilizatori se conecteaza. Acest model se numeste *near video on demand* deoarece filmul nu incepe exact la momentul asteptat ci cu o anumita intarziere.

Parametrul cheie al acestui model este durata intre doua transmisii. Daca se inepe o transmisie la fiecare 2 minute atunci sunt necesare 60 fluxuri pentru un film de 2 ore. Acest timp ramane la latitudinea operatorului si tine de cat de mult sunt dispusi utilizatorii sa astepte. Cu cat durata de asteptare este mai mare, cu atat sistemul este mai eficient. Acest model insa nu beneficiaza de de avantajele VCR, adica filmul nu mai poate fi oprit si repornit ulterior din acelasi moment. Utilizatorul va trebui sa astepte inceperea fluxului anterior si urmarirea pentru o perioada a unei parti a filmului deja vizualizata. Situatia ideala este o combinatie intre *near video on demand* (datorita eficientei) si control VCR pentru confortul utilizatorului. Solutia consta in folosirea unui buffer care inregistreaza ultimele *T* minute difuzate. Utilizatorul trebuie insa sa aiba posibilitatea de a citi doua fluxuri simultan.

## Distribuirea fisierelor

Fisierele multimedia au in general dimensiuni foarte mari, sunt scrise o singura data ,citite foarte des si accesate de cele mai multe ori secvential. Modul in care aceste fisiere sunt stocate poate fi imbunatatit fata de sistemele de operare obisnuite.

### Distribuirea unui singur fisier pe un singur disc

Pentru fisierele multimedia este esential sa se reduca numarul de cautari in interiorul aceluiasi fisier. O modalitate de a elimina acest neajuns este folosirea de fisiere coninue. Mentinerea continuitatii fisierelor multimedia este o sarcina dificila intrucat acestea pot contine imagini, sunet si text, stocate fiecare ca un fisier continuu. Pentru a se cauta un anume moment dintr-un film va trebui deci sa se caute acest moment si in fisierul audio, si in cel video si in subtitrare (text). Solutia consta in intreteserea imaginilor cu sunet si text, rezultand un nou format continuu.

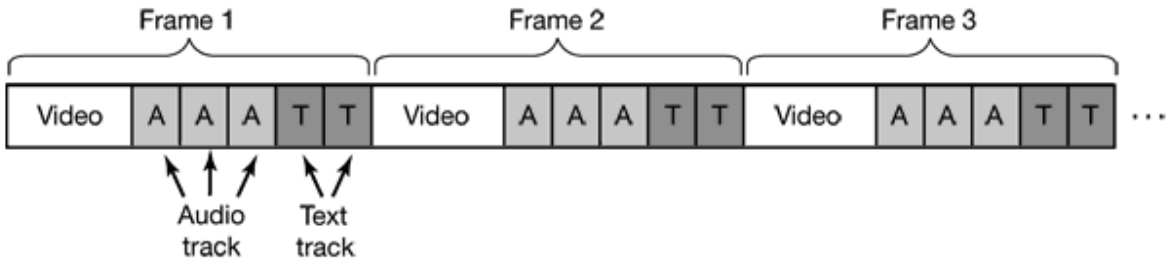


Fig. 3 Intreteserea fisierelor audio, video si text [1]

Acest format elimina necesitatea de a se cauta in 3 fisiere diferite insa creste cantitate de I/O pentru citire si necesarul de memorie din buffer. Pentru servere video cu fluxuri video concurente avantajul oferit de acest tip de fisiere continue este pierdut intrucat dupa citirea unui cadru din-un film vor fi necesare citiri si din alte fisiere video.

Exista mai multe alternative de organizare a fisierelor multimedia. Primul dintre acestea poarta numele de *modelul in blocuri mici (1-2 KB).* In acest mod de organizare blocurile discului se aleg mult mai mici decat marimea medie a unui cadru. Fiecare cadru contine informatii video, audio si text stocate in mod continuu pe disc. Se va creea un index pentru cadre, iar fiecare membru a acestei structuri va indica inceputul unui cadru. In acest mod, citirea unui cadru *k* consta in citirea indexului catre acest cadru pentru a se gasi si citi intregul cadru intr-o singura operatie.

O alta metoda este stocarea unui fisier multimedia in *blocuri mari* de date (Ex: 256 KB) fiecare bloc continand mai multe cadre. Este de asemenea necesar un index pentru aceste blocuri. Indecsii folositi sunt similar i-nodurilor din sistemele UNIX la care se mai adauga informatii referitoare la cadre.

In general, un bloc nu va contine un numar intreg de cadre. Pentru aceasta problema exista doua rezolvari.

- Daca un cadru nu incape intr-un bloc atunci cadrul va fi inclus in urmatorul bloc iar blocul curent va ramane cu o portiune goala. Aceasta solutie duce insa la fragmentarea spatiului.
- Daca un cadru nu incape intr-un bloc atunci acesta va fi divizat intre doua blocuri consecutive. Fragmentarea este redusa insa scade si performanta deoarece se vor face cautari suplimentare.

Daca se doreste sa se faca o alegere intre folosirea blocurilor mici sau a blocurilor mari de date trebuie sa se aiba in vedere urmatoarele considerente:

1. Index de cadre – foloseste mai mult RAM dar fragmentarea este mai redusa
2. Index de blocuri (fara se se imparta cadrele intre blocuri) – se foloseste mai putin RAM dar discul se fragmenteaza puternic
3. Index de blocuri (cu impartire intre cadre) – foloseste putin RAM; nu se fragmenteaza discul; sunt necesare cautari suplimentare

In toate cazurile insa se recomanda ca stocarea sa se faca in asa fel incat distanta intre blocurile de date sa nu fie mai mare de cativa cilindri pentru a reduce timpul de cautare pe cat posibil. Pentru a se obtine un mod de stocare cat mai compact, discul se divide in grupuri de cilindre pentru care se retine o lista a spatiilor goale (de 1KB, 2KB, .. etc.) astfel incat sa se incerce potrivirea cat mai multor blocuri in aceste gauri.

O alta diferenta intre cele doua abordari consta in modalitatea de *buffering*. Daca se folosesc blocuri mici, de fiecare data se va citi un singur cadru, deci un *buffer* care sa stocheze cadrul curent si cel anterior este suficient. *Bufferul* trebuie sa fie suficient de mare incat sa cuprinda orice cadru. Dimensiunea acestuia poate fi alocata fie dinamic in functie de marimea cadrelor, fie i se aloca dimensiuni fixe. Daca se folosesc blocuri mari este necesara o strategie mai complexa deoarece fiecare bloc contine mai multe cadre sau portiuni de cadre. Nu se mai poate folosi astfel un buffer dublu intrucat ar consuma foarte multa memorie si se opteaza in general catre un buffer circular de dimeniuni putin mai mari decat un bloc de transmisiune.

Un alt factor care diferentiaza cele doua metode de distributie este performanta de scriere si citire asupra discurilor. Daca se folosesc blocuri mari discurile vor functiona permanent la viteze mari si acest lucru poate conduce la stricarea acestora. Daca se folosesc blocuri mici in schimb, acestea nu vor putea fi distribuite pe mai multe discuri si performanta va scadea.

## Distribuirea mai multor fisiere pe un singur disc

De cele mai multe ori pe un server video exista mai multe fisiere nu numai unul singur. Daca acestea sunt imprastiate aleator pe suprafata discului ase va pierde mult timp pentru a se trece de la un fisier la altul atunci cand fisierele sunt accesate de diferiti utilizatori. Situatia poate fi depasita daca se tine o evidenta a preferintelor utilizatorilor catre anumite fisiere. Predictia se bazeaza pe *legea lui Zipf* care sustine ca probabilitatea ca un utilizator sa aleaga dintr-o lista ordonata de preferinte intrarea de ordin *k* este C/k, unde C este o constanta de normalizare. Asadar, daca lista ordonata de preferinte este o lista a celor mai populare filme, primele 3 vor avea probabilitatile de alegere: C/1 , C/2 respectiv C/3, unde C se alege astfel incat suma termenilor sa fie 1.

$$\frac{C}{1}+\frac{C}{2}+\frac{C}{3}+\cdots+\frac{C}{N}=1$$

Din aceasta relatie se poate calcula C in functie de N. Pentru un server video legea arata ca cel mai popular film are de doua ori mai multe sanse sa fie ales decat cel de-al doilea, de 3 ori mai mare decat cel de-al treilea samd. Avand aceasta informatie la dispozitie fisierele pot fi plasate astfel incat sa se creasca performanta sistemului. Strategia care se foloseste este simpla si independenta de distributia fisierelor si poarta numele de *algoritmul orga* (organ-pipe algorithm – Grossman, Silverman 1973). Acest algorithm plaseaza cel mai probabil fisier in centrul discului, al doilea cel mai probabil in jurul acestui fisier samd. Acest mod de distributie functioneaza mai eficient daca fisierele sunt continue. Numele algoritmului provine de la histograma distributiei fisierelor care se aseamana cu o orga.

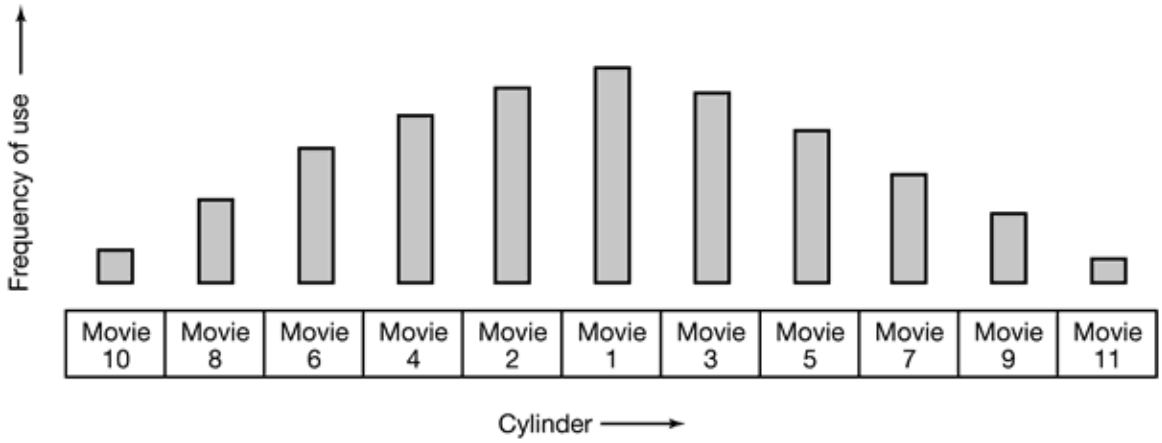


Fig. 4 Histograma distributiei fisierelor pentru algoritmul orga [1]

Algoritmul incearca sa mentina capatul de citire in centrul discului cat mai mult timp.

## Distribuirea fisierelor pe mai multe discuri

Pentru a se obtine performante mai bune serverele video folosesc de cele mai multe ori mai multe discuri care ruleaza in paralel. Sistemele RAID nu reprezinta tot timpul o solutie intrucat securitatea datelor nu este atat de importanta pe cat este performanta. De cele mai multe ori se foloseste o configuratie formata dintr-un numar mare de discuri numita *ferma de discuri.* Discurile nu se rotesc sincronizat si nu contin informatii de paritate ca RAID. Filmele sunt stocate pe cate un disc si in caz de defectiuni pot fi recopiate de pe alte medii de stocare. Dezavantajul acestei configuratii este ca nu se echilibreaza incarcarea pe discuri, unele fiind accesate mult mai des decat altele. Cu alte cuvinte, sistemul nu va fi utilizat complet pana cand nu se va face o evidenta a accesarii fisierelor.

O alta configuratie ce se poate folosi este de a imparti un fisier pe mai multe discuri. Pe fiecare disc se vor scrie blocuri de dimensiuni egale. Nici aceasta metoda nu elimina problema incarcarii echilibrate a discurilor intrucat intotdeauna primul disc va trebui sa fie accesat. Se prefera asadar o distributie aleatoare a datelor pe discuri. O alta problema care se pune este dimensiunea blocurilor care se scriu pe fiecare disc. Daca se dispune de un numar mare de discuri fisierul se poate imparti astfel incat pe fiecare disc sa se scrie o singura parte din fisier astfel incat fiecare disc sa fie accesat doar odata. A doua varianta este de a imparti discurile in grupuri si de a scrie fiecare fisier pe o partitie. Prima varianta ofera o performanta mai buna insa in caz de defectare a unui disc fisierul nu va mai putea fi accesat. A doua varianta nu echilibreaza la fel de bine incarcarea discurilor insa ofera o mai buna toleranta la erori sau probleme tehnice.

## Caching

Modelele clasice de caching pentru multimedia (LRU, FIFO, etc. ) nu functioneaza eficient pentru fisierele multimedia. In modelele clasice, blocurile folosite frecvent sunt tinute pentru o perioada deoarece exista sanse ca acestea sa fie refolosite. La sistemele multimedia modelul de acces este secvential: un fisier multimedia este vizualizat de la un capat la celalalt fara a se reveni de foarte multe ori la momente anterioare. Asadar un bloc are sanse mici sa fie refolosit. Cachingul nu mai este eficient in aceste situatii insa poate fi exploatat in alte moduri.

Spre exemplu, daca doi utilizatori vizualizeaza acelasi film, dar au inceput la 2 secunde unul de celalalt, exista o sansa foarte mare ca cel care a inceput mai tarziu vizualizarea sa vada exact aceleasi cadre ca si primul utilizator. Sistemul de operare trebuie sa tina evidenta filmelor care sunt vizualizate doar de unul sau de mai multi utilizatori. Asadar, atunci cand exista posibilitatea ca un cadru sa fie redat la intervale scurte de timp, atunci cachingul are sens.

O alta varianta de folosire a cachingului este de a contopi doua fluxuri intr-unul. Daca doua filme sunt redate cu un decalaj suficient de mare astfel incat cachingul sa nu fie fezabil, se pot modifica ratele de redare astfel incat filmele sa apara ca fiind sincronizate. Acest lucru este posibil prin memorarea in cache a ambelor rate de redare.

Cachingul se poate folosi de asemenea pentru a stoca primele minute dintr-un film astfel incat acestea sa fie permanent disponibile utilizatorilor, iar intre timp restul filmului sa fie copiat de pe alte medii de stocare.

## Planificarea stocarii fisierelor multimedia

In sistemele multimedia accentul cade pe rate de date mari si livrare in timp real. Ambele cerinte sunt dificil de indeplinit fara anumite optimizari specifice sistemelor de operare multimedia. Pe langa modul de distribuire al datelor pe discuri este de asemenea necesara si o anumit rutina de accesare a acestora. Ca si la procese acest lucru se realizeaza cu ajutorul unui planificator.

### Planificarea statica

Cantitatea mare de date si rata de date in timp real pe care sistemele multimedia le necesita pot fi compensate cu ajutorul unei trasaturi fundamentale a acestora: previzibilitatea. In sistemele de operare clasice, cererile catre disc sunt efectuate intr-o maniera impredictibila. Discul nu poate decat sa astepte sa primeasca o cerere dupa care sa verifice daca blocul exista dupa care il localizeaza si il citeste. In

sistemele multimedia acest lucru este diferit intrucat fiecare flux de date activ incarca sistemul doar cu o cantitate deja cunoscuta. Spre exemplu, pentru o redare NTSC, la fiecare 33.3 ms fiecare client asteapta cadrul urmator deci sistemul va avea 33.3 ms la dispozitie pentru a-l obtine. Aceasta incarcare predictiva poate fi folosita pentru a planifica modul de acces la discuri.

Se va considera in continuare ca se foloseste doar un disc insa tehnicile pot fi aplicate similar si mai multor discuri. Se va considera ca exemplu ca exista 10 utilizatori, fiecare vizualizand un film diferit. In functie de sistemul de operare, calculatorul va avea 10 procese (cate unul pentru fiecare film) sau un proces cu 10 threaduri. Timpul este divizat in runde, unde o runda reprezinta durata unui cadru. La inceputul fiecarei runde se genereaza o cerere de acces la disc din partea fiecarui utilizator. Dupa ce toate cererile au sosit, discul stie ce are de facut in acea runda. De asemnea stie ca nu vor mai sosi alte cereri pana cand nu va incepe runda urmatoare. In consecinta poate sorta cererile intr-un mod optim (in ordinea cilindrilor) si le va procesa de asemenea intr-o ordine optima.

La prima vedere aceasta planificare pare ca nu aduce nicio imbunatatire deoarece atat timp cat sarcinile sunt efectuate inainte de timp nu se obtine nici o crestere de performanta. Totusi, daca sarcinile se finalizeaza mai rapid, timpul ramas poate fi folosit pentru procesarea altor fluxuri. Pentru a mentine fluxul de date se foloseste de asemenea *bufferingul*. In timpul unei runde se foloseste un set de buffere, cate unul pentru fiecare flux. Cand runda se finalizeaza, procesele de iesire sunt deblocate si se incepe transmiterea cadrului. In acelasi timp trebuie efectuate alte cereri folosind un al doilea set de buffere, in timp ce primul set este liber si poate trece la cadrul urmator. Limitarea la o runda pentru fiecare cadru nu este obligatorie fiind doar o simplificare. Daca se doreste reducerea spatilui ocupat de buffer se pot folosi mai multe cadre intr-o runda. Strategia care se foloseste depinde de posibilatile de memorare sau de viteze de I/O a discurilor.

## Planificarea dinamica

Planificarea dinamica este necesara atunci cand ratele de redare si rezolutiile difera. In aceste situatii cererile de acces la discuri apar intr-un mod aleator. Fiecare cerere specifica blocul care se doreste a fi citit si termenul pana la care acest bloc trebuie sa fie obtinut. Se presupune ca timpul de procesare pentru fiecare cerere este fix, astfel incat planificatorul trebuie sa tina cont doar de termene.

Cand sistemul porneste nu exista cereri de acces la disc. Prima cerere primita este procesata imediat si in timp ce are loc cautarea cererile care ajung trebuiesc si ele procesate. Este insa necesar un algoritm pentru a se alege o anumita cerere daca exista concurenta. Pentru a se decide se iau in considerare doi factori: termenul si cilindrii. Din punct de vedere al performantei, daca cererile sunt sortate pe cilindri timpul de cautare este redus insa unele cereri s-ar putea sa isi depaseasca termenul de finalizare. Din punct de vedere al performantei in timp-real este mai eficient sa se sorteze cererile in functie de termenul de finalizare insa acest lucru poate conduce la cresterea timpului de cautare. Pentru un echilibru intre cele doua considerente se foloseste algoritmul *scand-EDF*. Ideea care sta la baza acestui algoritm este de a grupa cererile care au termene apropiate si sa le proceseze in ordinea cilindrilor.

Cand fluxuri diferite au rate da date diferite apare o noua problema atunci cand apare un nou utilizator. Daca acceptarea clientului va provoca depasirea termenelor pentru cererile existente atunci acest client nu va fi acceptat. Exista doua modalitati de a calcula de cate resurse noi are nevoie un nou client. (latime de banda, memorie, capacitate de procesare). Daca resursele sunt suficiente pentru un client mediu, atunci acesta este acceptat.

## Concluzii

- Sistemele de operare multimedia contin optimizari pentru redarea datelor in timp real
- Pentru optimizarea sistemelor se folosesc particularitatile fisierelor multimedia
- Sistemele de operare multimedia pot fi construite pe baza sistemelor de operare obisnuite
- Este importanta stocarea continua a fisierelor
- Modul de amplasare a datelor afecteaza performantele de redare

## Bibliografie

[1] Andrew Tanenbaum : Modern Operating  Systems – Third Edition 2009– Prentice Hall, Upper Saddle River, NJ

[2] Daniel Alexander Taranovsky -CPU Scheduling in Multimedia Operating Systems

[3] Silberschatz, Galvin and Gagne Operating System Concepts - 7th Edition, Jan 2