IKER GAZTE
NAZIOARTEKO IKERKETA EUSKARAZ

IV. IKERGAZTE
NAZIOARTEKO IKERKETA EUSKARAZ

2021eko ekainaren 9, 10 eta 11a
Gasteiz, Euskal Herria

ANTOLATZAILEA:
Udako Euskal Unibertsitatea (UEU)

INGENIARITZA ETA ARKITEKTURA

Partizioen konfigurazioaren eragina denbora errealeko sistemen planifikagarritasunean

Andoni Amurrio

169-176 or.
https://dx.doi.org/10.26876/ikergazte.iv.03.21

Partizioen konfigurazioaren eragina denbora errealeko sistemen planifikagarritasunean

Amurrio González, A.[1]

Ikerlan (BRTA). Sistema Txertatu Fidagarriak, J. M. Arizmendiarrieta Pasalekua, 2 - 20500 Arrasate-Mondragón (Gipuzkoa)

aamurrio@ikerlan.es

Laburpena

Denbora errealeko sistemen garapenerako ezinbestekoa da diseinu faseetan hartzen diren erabakiek daukaten eragina ezagutzea. Partizioetan oinarritutako arkitekturak gero eta gehiago erabiltzen dira segurtasun funtzionaleko baldintzak dituzten aplikazioak diseinatzeko, eta euren konfigurazioak zuzeneko eragina du sistemen planifikagarritasunean. Hori dela eta, lan honetan partizioak inplementatzeko testuinguru-aldaketa denbora aintzat hartzen duen sistema eredu bat erakutsiko da, eta ondoren diseinu faseetan ezartzen diren konfigurazio parametroen eraginak ebaluatuko dira. Azterketa horren ondorioak planifikazio algoritmo heuristikoen garapenerako irizpide gisa erabiltzeko baliagarriak izango dira.

Hitz gakoak: Denbora errealeko sistemak, Planifikagarritasuna, Partizioak

Abstract

It is necessary to know the effect of design-choices taken during the development of real-time systems. Partition-based architectures are more and more used in the design of safety critical applications, and their configuration has direct impact in the schedulability of the system. That is why in this work, a system model that considers context switch overheads is shown, later to evaluate the effects of configuration parameters that are set during design phases. The conclusions drawn from this study may serve as a guide for the development of heuristic scheduling algorithms.

Keywords: *Real-Time Systems, Schedulability, Partitions*

1. Sarrera eta motibazioa

Denbora errealeko sistemetan, sistema ziberfisikoek (konputazio eta komunikazio elementuak bateratzen dituzten sistema informatikoek) konputatutako emaitzak ez dira bakarrik euren balioen zuzentasunean oinarritzen, baizik eta lorturiko emaitzak zuzenak izateaz gain, aurretik inposaturiko epe baten barruan lortu izana aintzat hartu beharra dago (Stankovic, 1988). Adibidez, industrian edo garraioetan inplementatzen diren kontrol aplikazioek epe zorrotz batzuk bete behar dituzte, eta ez betetzekotan ondorio arriskutsuak suerta daitezke. Kasurik txarrenean ere epeak betetzen direla bermatzeko, planifikazio teknikak erabiltzen dira. Oro har, sistema ziberfisikoak osatzen dituzten prozesadoreetako atazak noiz exekutatzen diren zehazteari planifikazioa deritzo, eta kasu ez tribialetan problema konplexu bat da (Stallings *et al.*, 2008). Planifikazio metodo ezagunenak exekutibo ziklikoa, lehentasunetan oinarritutakoa edota bien konbinazioa den planifikazio hierarkikoa dira, besteak beste.

Industria zein garraio domeinuetan maiz segurtasun funtzionaleko arauak bete behar dira, kontrol aplikazioek errore maila onargarriak eduki dezaten eta horiek suertatzerakoan behar beste neurri aplikatzeko. Aplikazioek bermatzen duten segurtatze-mailaren arabera *Safety Integrity Level* (SIL) izeneko mailak definitzen dira IEC 61508 estandarrean, eta SIL maila ezberdineko konponenteak exekuzio plataforma berdinean exekutatu behar badira, euren arteko isolamendu egokia bermatu beharra dago (IEC, 2010). Horretarako, partizioak erabiltzen dira. Denbora isolamendua bermatzeko, denbora partizioak inplementatzeko gai diren sistema eragileak erabiltzen dira, hala nola Integrity (Green Hill Software). Atazak partizioetan kokatzean, partizioen exekuzioa ere planifikatu beharra dago, sistema osoaren erantzun- denbora, hots ataza guztiek euren betebeharrak amaitzeko behar duten denbora totala, inposaturiko epea baino luzeagoa izan ez dadin.

Denbora errealeko sistemen planifikazioa aspladi jorraturiko gai bat da. Izan ere, lehengo planifikazio eta analisi lanak 1970.eko hamarkadakoak dira (Liu eta Layland, 1973). Hala ere, exekuzio paradigmak eta sistemen konplexutasuna denboraz aldatu eta garatu diren heinean, planifikazio algoritmoek ere sistema eredu horietara egokitu behar izan dute. (Amurrio *et al.*, 2019) lanean azken 40 urtetan denbora errealeko sistemen planifikazioa era batean edo bestean jorratzen duten lanak bildu ziren, eta ondorio garrantzitsuenetariko bat sistema partizionatuen planifikazioa oso lan gutxitan jorratu dela nabarmendu zuten autoreek.

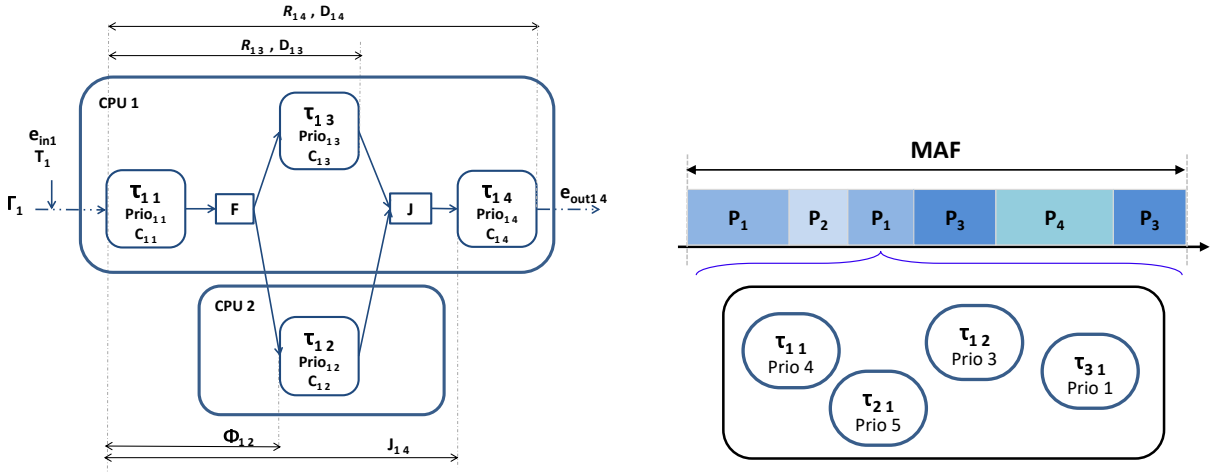
Lan honen helburua trenen gidatzea kontrolatzen duten aplikazioen planifikazioan ardaztuko da. Izan ere, gaur egun aplikazioak exekutibo ziklikoetan oinarritzen dira: aplikazioaren funtzio guztiak ziklikoki exekutatzen dira, nahiz eta beti lan baliagarririk egiten ez duten. Gainera, exekuzio paradigma hori erabilita zaila da konponenteen arteko isolamendua bermatzea. Hortaz, trenen kontrol aplikazioen ber-diseinu orokor bat burutzea proposatu zen (Fang eta Obermaisser, 2017) lanean, partizioetan oinarritutako arkitekturak eta planifikazio hierarkikoa erabilita. Beraz, partizioetan oinarritutako denbora errealeko sistemak planifikatzeko algoritmo bat garatzeko bidean, partizioen konfigurazioaren parte diren parametro jakin batzuek sistemaren planifikagarritasunean duten eragina aztertu beharra dago.

2. Sistema Eredua

Sistema baten ezaugarriak edota portaerak ulertzeko, iragartzeko eta kontrolatzeko egiten diren errepresentazio sinplifikatuei sistema ereduak deitzen zaie. Lan honetan Kantabriako Unibertsitatean garaturiko MAST (Modelling and Analysis Suite for Real-Time applications) [1] ereduarekin bat datorren sistema eredu bat eraikiko da, segurtasun aplikazioen denbora portaera deskribatzeko.

2.1. Arkitektura Logikoa

Ereduaren elementu nagusia gertaera jakin bati erantzunez aktibatzen den *End-to-End Flow* (*e2e flow* aurrerantzean) izeneko jarduera katea da, eta 1(a) irudian adibide sinple bat irudikatu da. Jarduera bakoitza prozesadore batean exekutatzen den ataza bat da, eta *step* izeneko elementuaren bitartez adierazten da. Aurrerantzean *step* eta ataza terminoak berdin erabiliko dira. Aplikazioen exekuzioa aktibatzen dituen gertaerei *Event* (e_in) deitzen zaie eta T_i periodoa dute. Hortaz, Γ_i *e2e flow*-ko j-garren atazari τ_{ij} deituko zaio, eta kasurik txarreneko exekuzio denbora jakin bat dauka, C_{ij}, eta lehentasun bat, $Prio_{ij}$. Atazek errepresentatzen duten prozesadorearen erabilera denbora erlatiboari U_{ij} ditzen zaio, kasurik txarreneko exekuzio denboraren eta aktibazio periodoaren arteko erlazioa delarik: $U_{ij} = C_{ij}/T_i$. Euren exekuzio denborarekin alderatuz askoz txikiagoa denez, lan honetan ez da kontuan hartuko atazen testuingurua kargatzeko behar den denbora, hau da, ataza mailako testuinguru-aldaketa.



(a) Arkitektura logikoa (*e2e flow* banatua)

(b) Partizio eredua

1. Irudia: Sistema eredua

Lehen esan bezala, aplikazioen exekuzioari epe bat inposatzen zaio, beraz ataza bakoitzak D_{ij} izeneko epe

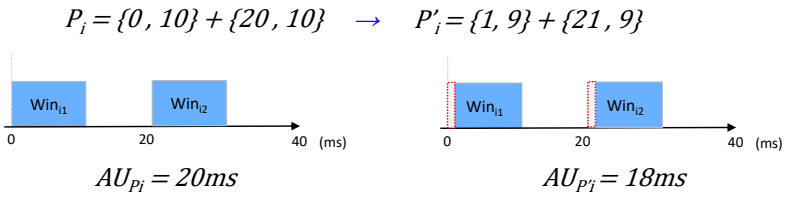
[1] https://mast.unican.es/

bat eduki dezake. Sistemen erantzun-denbora kasurik txarrenean euren exekuzioa burutzeko atazek behar duten denbora totala da, gainerako atazek eta beste edozein elementuren interferentzia guztiak aintzat hartuta. Ataza bakoitzaren erantzun-denbora R_{ij} da eta, nahitaez, inposaturiko epea baino motzagoa izan behar du.

2.2. Partizio eredua

Prozesadore bakoitzean, atazak P_x partizioetan kokatzen dira. Partizioak periodikoki exekutatzen dira exekutibo zikliko (inglesez $MajorFrame$ edo MAF) baten barruan, eta k denbora leihoz (Win_{xk}) konposatuta egon daitezke, 1(b) irudian adierazten den moduan. Partizio leihoen hasiera uneari S_{xk} deritzo, eta denbora luzerari L_{xk}. Partizio bakoitzaren erabilera bertan kokatutako atazen erabilera erlatiboen batuketa da, hots, $U_{P_x} = \sum_{\forall \tau_{ij} \in P_x} U_{ij}$. Partizio bakoitzari prozesamendu denbora totaleko zati bat esleitzen zaio, AU_{Px}.

Oro har, prozesadoreek denbora tarte bat behar dute partizio leiho bakoitzaren testuingurua kargatzeko, hots, prozesamendurako erabili ezin den denbora jakin bat dago. Sistema eragilearen eta exekuzio-arkitekturaren araberakoa da, eta exekuzio osoan zehar konstante mantentzen dela suposatuko dugu. Partizio-mailako testuinguru-aldaketa denbora horri CS deituko zaio eta lan honetan partizio lehio bakoitzaren hasieran ematen dela suposatuko dugu, 2. irudian erakusten den bezala. Beraz, partizio leiho bat deskribatzeko honako notazioa erabiliko da: $Win_{xk} = \{ S_{xk} + CS , L_{xk} - CS\}$.



2. Irudia: Testuinguru-aldaketa denboraren eragina partizio leihoetan ($CS = 1$)

2.3. Planifikagarritasun analisia

Denbora errealeko sistemen erantzun-denbora planifikagarritasun analisi izeneko metodo matematikoen kalkulatzen da, eta sistema eredura egokituta egon behar dute kalkulatutako emaitza benetazko kasurik txarrenekoa izan dadin. Lehen analisia (Liu eta Layland, 1973) lanean aurkeztu zen aspaldi, ataza independiente eta periodikoen planafikagarritasuna aztertzeko. Handik aurrera, analisi teknika asko sortu dira, eta gure sistema ereduarekin antzekotasun gehien dutenak hauexek dira:

- (J.J. Gutiérrez, J.C. Palencia and M. González Harbour, 2000) lanean egileek *Holistic* teknika (Audsley *et al.*, 1991) *e2e flow* ez linealetara aplikatzea porposatu zuten. Hala ere, teknika hori oso pesimista dela aski ezaguna da denbora errealeko ikerketa komunitatean.
- (Palencia eta González Harbour, 1998) laneko autoreek *Offset-based* teknika garatu zuten, *Holistic* teknikaren pesimismoa gutxitzeko, eta sistema partizionatuk analizatzeko gai dena (Palencia *et al.*, 2016). Aldiz, *e2e flow* linealak baino ez zituzten aintzat hartu.

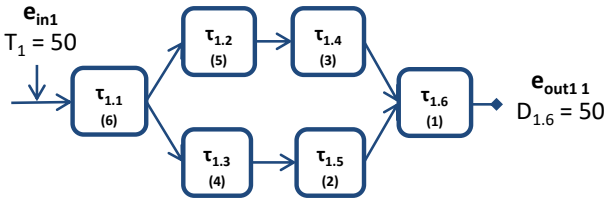
Horregatik (Amurrio *et al.*, 2020) lanean *e2e flow* ez linealen eta partizionatutako sistemen planifikagarritasuna analizatzeko proposamena egin zen. Bertan, *Holistic* teknikaren pesimismoa gutxitzen dela baieztatu zen, baita partizioetan oinarritutako sistemetara aplikagarria dela ere bai. Gure azterketa eta esperimentu guztiak analisi teknika horretan oinarrituta daude.

3. Partizioen konfigurazioaren eragina aztertzen

Atal honetan partizioen konfigurazio parametro desberdinen eragina aztertuko da. Horretarako, deskribatutako ereduarekin bat datorren eta segurtasun aplikazioen adierazgarri den adibide sinple bat erakutsiko da, hurrengo esperimentuak kasu sintetiko horren inguruan jorratzeko.

3.1. Aplikazio sintetikoa

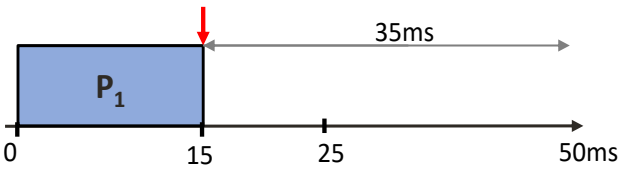
Segurtasun aplikazioen adibide izango den aplikazio sintetiko bat 3. irudian errepresentatu da. Aplikazio sintetikoak ohiko segurtasun aplikazioen ezaugarri nagusiak ditu, hala nola aktibazio periodo jakin bat, prezendentzia erlazioak dituzten atazen exekuzioa eta kasurik txarrenean ere bete beharreko epe bat. 50ms-ro aktibatzen den eta sei *step*-ez konposaturiko *e2e flow* ez lineal bat da, prozesadore bakar batean mapeatu da eta prozesadorean partizio bakar batean (P_1) kokatu da. Sinpletasuna bermatzeko, ataza guztien exekuzio denbora 2ms direla suposatuko da, eta esleitzen zaien lehentasuna parentesi artean idatzi da. Hortaz, aplikazioaren erabilera erlatibo totala %24koa da. Aplikazio sintetiko honen erantzun-denborari buruz ari garenean, 6. *step*-aren erantzun-denboraz (R_{16}) ari gara, horixe baita aplikazioaren azken ataza. Bestalde, esperimentu guztietarako erabiliko den MAF periodikoaren balioa 50ms izango da.



3. Irudia: Aplikazio sintetikoa

3.2. Partizioen analisia eta konfigurazioa

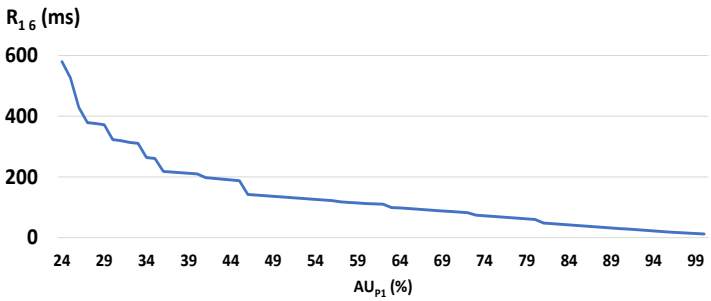
Denbora errealeko sistemen erantzun-denborak kalkulatzen dituzten algoritmoak ezkorrak edo pesimistak direla esaten da. Izan ere, egoerarik txarrena haintzat hartu behar dute beti kalkulu guztietan. Demagun aplikazio sintetikoaren parte den 2ms-ko ataza bakar baten erantzun-denbora kalkulatu nahi dela, 4. irudian adierazi den partizioan kokatu delarik. Adibide horretan, kasurik ezkorrena tarearen aktibazio ebentua partizioaren amaieran suertatzea litzateke (gezi gorri batez irudikatuta). Hortaz, tarearen erantzun-denbora kalkulatzeko, bi termino hartuko dira kontutan: batetik tarearen exekuzio denbora, 2ms, eta bestetik, tareak prozesadorea erabiltzea galarazita duen denbora tartea, 35ms (gogoratu MAFaren aktibazioa ere periodikoa dela). Beraz, atazaren kasurik txarreneko erantzun-denbora 37ms izango litzateke. Agerikoa da partizioari zenbat eta exekuzio-denbora gehiago esleitu orduan eta txikiagoak izango direla barneko tareen erantzun-denborak.



4. Irudia: P_1 partizioan dagoen tarea aktibatzen duen ebentua partizioaren amaieran jazo da, eta MAFaren zikloa berriro hasi arte ezin da exekutatu

Ondorioz, aplikazioen garapen fase goiztiarretan hartu beharreko erabaki bat, partizio bakoitzaren exekuziorako esleitzen zaion denbora da. Hots, lehen deskribatutako ereduaren arabera eta lantzen ari garen adibide sintetikoaren arabera, partizioaren AU_{P1} parametroa finkatu beharra dago. Izan ere, kasurik txarreneko erantzun-denboraren kalkuluan eragin zuzena dauka, 5. irudian azaltzen den bezala. AU_{P1} parametroaren balio minimoa (partizioari esleituriko prozesamendu denbora minimoa) bertan dauden tareen exekuzio denboraren araberakoa izango da. 3. irudiko aplikazio sintetikoaren sei tareak P_1 partizioan kokatu dira, eta euren prozesadore-erabilera erlatiboen batura (U_{ij} sistema ereduan) %24-koa denez, horixe izango da P_1 partizioari esleituko zaion MAFaren ehuneko minimoa. Aipagarria da erantzun-denborak zenbateraino murriztu daitezken partizioari esleitutako prozesamendu-denboraren arabera: 580ms-tik, prozesamendu-denbora minimoa esleituz, 12ms-ra, prozesadorearen %100 esleituz.

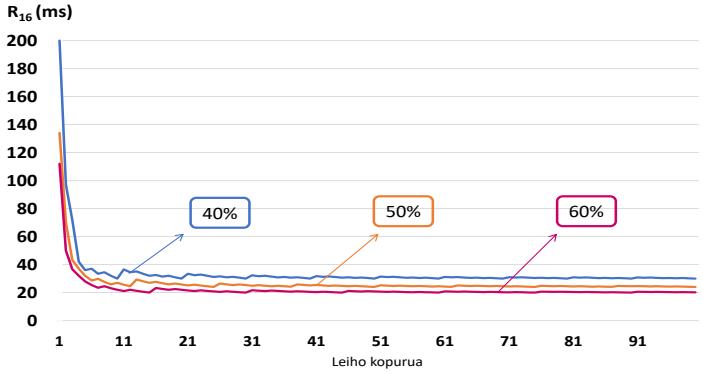
Lehen azaldutako analisi tekniken ezkortasuna gainditzeko beste modu bat partizio leiho kopurua handitzean datza, partizioari esleitutako prozesamendu denbora konstante mantentzen delarik. Izan ere, suerta daiteke AU_{P1}



5. Irudia: erantzun-denboraren eboluzioa AU_{P_1} parametroaren menpe

parametroa handitzea galarazita egotea, adibidez bestelako partizioek (komunikazioetarako, kritikotasun maila ezberdineko beste aplikazio bat exekutatzeko...) MAFaren zati handiago bat beharrez gero. Leiho kopurua handituz, exekuzioa galarazita dagoen denbora tarteen tamaina murrizten da, erantzun denborak era berean txikiagotuz. 6. irudian aplikazioaren erantzun-denboraren eboluzioa leiho kopuruarekiko ikusten da, hasieran esleitutako denbora total ezberdinetarako (%40,%50 eta %60). Adibide guztietan leihoak uniformeki banatu dira MAFean zehar.

Oro har, partizio leiho kopuruak gora egiten duenean, erantzun-denboraren txikiagotzea nabarmena da. Orain arteko esperimentuetan ez da kontutan eduki lehen azaldutako partizio mailako testuinguru-aldaketa denborarik. Prozesamendu denbora horren balioa aplikazioak exekutatzeko erabiltzen den sistema eragilearen araberakoa da. Testuinguru-aldaketa denborak neurtzen dituzten arloko egoerako lanek (Hamelin *et al.*) (Masmano *et al.*), 17μs eta 27μs bitarteko balioak erakusten dituzte, beraz gure saiakuntzetarako $CS = 20\mu$s-ko balioa erabiliko dugu, eta partizio-mailako testuinguru-aldaketa motelagoak irudikatzeko $CS = 200\mu$s-ko balioa ere aztertuko dugu.



6. Irudia: erantzun-denboraren eboluzioa partizio leiho kopurua handitzean, AU_{P_1}-ren balio ezberdinetarako

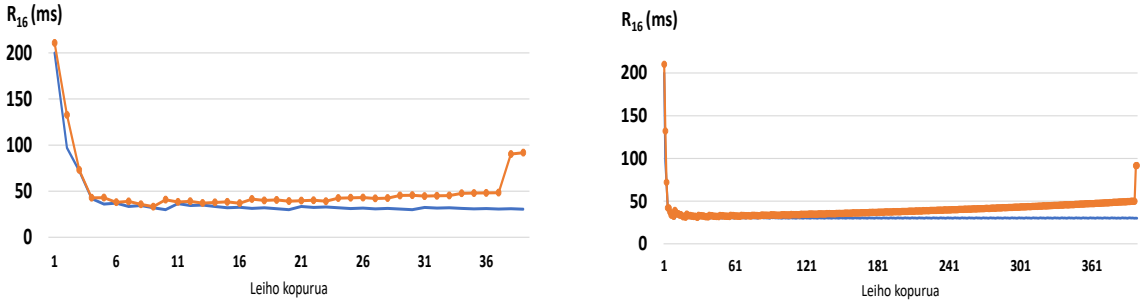
Partizioari esleituriko denbora totala eta testuinguru-aldaketarako denbora zein den jakinda, konfiguratu daitezken partizio leiho kopuru maximoaren (NW_{P1}) kalkulua halaxe da:

$$NW_{P_1} = \lfloor (AU_{P_1} - U_{P_1}) * \frac{MAF}{CS} \rfloor \quad (1)$$

(7.-9.) irudietan partizio konfigurazio desberdinetako kasurik txarreneko erantzun-denborak ikusi daitezke. AU_{P1}-ren balio desberdinetarako erantzun-denborak kalkulatu dira, partizio leiho kopurua 1etik NW_{P_1}-ra. Aztertutako prozesamendu denbora erabilera balioak lehengo berdinak dira, hots, %40 7.irudian, %50 8. irudian eta %60 9. irudian).

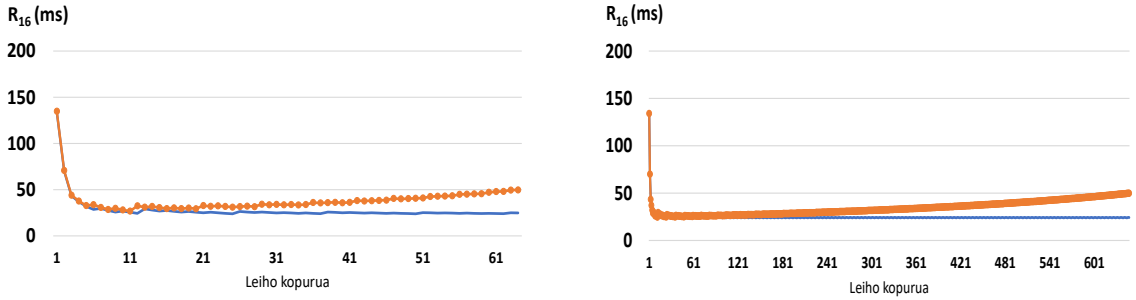
Kualitatiboki, AU_{P1}-ren balioa edozein dela, erantzun-denboraren eboluzioa berdina da beti: balio maximoa partizio leiho bakarra erabiltzen denean lortzen da, eta gutxituz doa kopurua igotzerakoan. Balio minimoetara heltzen denean, bi portaera nabarmendu behar dira. Batetik, egoera ideala suposatzen bada eta testuinguru-aldaketa denborak ez dira aintzat hartzen (lerro urdinak, $CS = 0$), erantzun-denbora egonkor mantentzen da partizio leiho kopurua handitzean. Bestetik, testuinguru-aldaketa denborak aintzat hartzen dituen analisian (lerro laranjak, $CS >$

0) erantzun-denborak gora egiten hasten dira leiho kopurua igotzerakoan. Izan ere, prozesamendurako erabilgarri ez dagoen denbora gero eta handiagoa da partizio leiho kopurua handitzen bada, eta lehen azaldu den bezala, horrek erantzun-denborak nabarmen igotzea dakar.



(a) $CS = 200\mu$s (b) $CS = 20\mu$s

7. Irudia: Kasurik txarreneko erantzun-denborak partizio leiho kopuruaren arabera - $AU_{P_1} = 40\%$

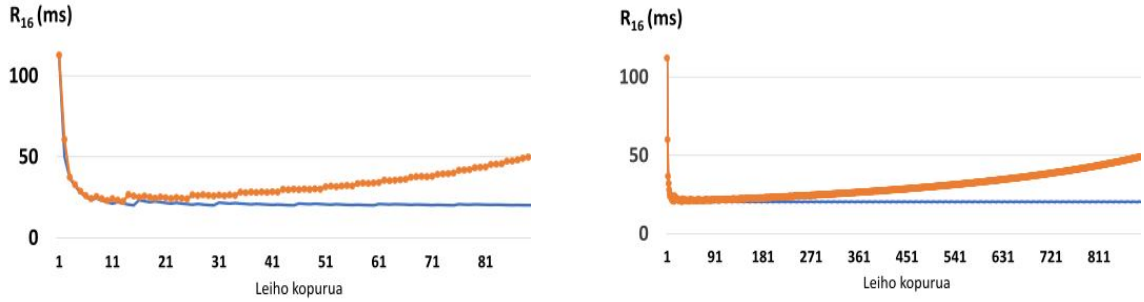


(a) $CS = 200\mu$s (b) $CS = 20\mu$s

8. Irudia: Kasurik txarreneko erantzun-denborak partizio leiho kopuruaren arabera - $AU_{P_1} = 50\%$

4. Ondorioak

Artikulu honetan partizioetan oinarritutako denbora errealeko sistemen planifikagarritasuna ebaluatu dugu, hainbat konfigurazio parametrok duten eragina aztertuz. Proposatutako sistema ereduak partizioen testuinguru-aldaketa denbora aintzat hartzen du, eta kasurik txarreneko erantzun-denbora kalkulatzean eragin garrantzitsua dutela baieztatu da. Gero, erantzun-denboraren kalkuluan partizioek eskuragarri duten denborak nola eragiten duen ebaluatu da, bai eta denbora hori leiho kopuru ezberdinetan banatzearen araberako efektua ere. Ondorio orokorra partizio



(a) $CS = 200\mu$s (b) $CS = 20\mu$s

9. Irudia: Kasurik txarreneko erantzun-denborak partizio leiho kopuruaren arabera - $AU_{P_1} = 60\%$

leihoak gehitzea erantzun-denbora murriztea dakarrela dela esan daiteke. Testuinguru-aldaketa denboraren eraginez kasurik txarreneko erantzun-denborak berriz gora egiten duenez, leiho kopuru aproposa finkatu beharra dago aplikazioaren epea errespetatuko dela bermatzeko.

5. Etorkizunerako planteatzen den norabidea

Lan honetatik abiatuta partizioen planifikazio algoritmo bat garatuko da, eginiko azterketaren ondorioak aintzat hartuta. Sistemaren konplexutasuna dela eta, baliteke pseudo-ausazko bilaketa algoritmo bat garatu behar izana, hala nola algoritmo genetiko bat (Kuijpers, 1996). Hala ere, lehenbizi heuristiko dedikatu batekin probatzea komenigarria da, lan honen bidez lortutako irakaspenak kasu gehienetara orokortu daitezkela ziurtatzeko. Bestalde, sistema ereduan aipatu diren zenbait planifikazio parametroren esleipena, hala nola lehentasunak, sarrerako parametro gisa erabili dira eta ez da haien eragina ebaluatu. Mota honetako azterketa bat egin daiteken arren, lehentasunak esleitzea bilaketa algortimoen betebehar bezala definitzea ere planteatu daiteke. Hala eta guztiz ere, kontuan hartu beharra dago bilaketa algoritmoak erabaki gehiago hartzera behartzean, gero eta motelagoak izango direla emaitzak konputatzeko, eta balizko soluzio optimoetara heltzeko probabilitatea nabarmen murrizten dela.

6. Erreferentziak

Amurrio, A., E. Azketa, J. J. Gutierrez, M. Aldea, eta M. G. Harbour. 2020. Response-time analysis of multipath flows in hierarchically-scheduled time-partitioned distributed real-time systems. *IEEE Access* 8.196700–196711.

Amurrio, Andoni, Ekain Azketa, J Javier Gutierrez, Mario Aldea, eta Jorge Parra. 2019. A review on optimization techniques for the deployment and scheduling of distributed real-time systems. *Revista Iberoamericana de Automática e Informática Industrial (in Spanish)* 16.249–263.

Audsley, Neil C, Alan Burns, Mike F Richardson, eta Andy J Wellings. 1991. Hard real-time scheduling: The deadline-monotonic approach. *IFAC Proceedings Volumes* 24.127–132.

Fang, Hongjie, eta Roman Obermaisser. 2017. Execution environment for mixed-criticality train applications based on an integrated architecture. In *2017 International Conference on Promising Electronic Technologies (ICPET)*, 1–7. IEEE.

Green Hill Software. Integrity RTOS.

Hamelin, Etienne, Moha Ait Hmid, Amine Naji, eta Yves Mouafo-Tchinda. Selection and evaluation of an embedded hypervisor: application to an automotive platform. Proceedings of the 10th Embedded Real-Time Systems International Congress (ERTS 2020).

IEC. 2010. IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems Part 1: General requirements.

J.J. Gutiérrez, J.C. Palencia and M. González Harbour. 2000. Schedulability analysis of distributed hard real-time systems with multiple-event synchronization. In *Proceedings 12th Euromicro Conference on Real-Time Systems. Euromicro RTS 2000*, 15–24. IEEE.

Kuijpers, Cindy. 1996. Algoritmo genetikoak saltzaile ibiltariaren probleman: Gipuzkoako bira egokiaren atzetik. *Elhuyar* 22.10–30.

Liu, Chung Laung, eta James W Layland. 1973. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)* 20.46–61.

Masmano, Miguel, Ismael Ripoll, Alfons Crespo, eta J Metge. Xtratum: a hypervisor for safety critical embedded systems. Proceedings of the 11th Real-Time Linux Workshop 2009, pages 263-272.

Palencia, J Carlos, Michael González Harbour, J Javier Gutiérrez, eta Juan M Rivas. 2016. Response-time analysis in hierarchically-scheduled time-partitioned distributed systems. *IEEE Transactions on Parallel and Distributed Systems* 28.2017–2030.

Palencia, Jose Carlos, eta Michael González Harbour. 1998. Schedulability analysis for tasks with static and dynamic offsets. In *Proceedings 19th IEEE Real-Time Systems Symposium (Cat. No. 98CB36279)*, 26–37. IEEE.

Stallings, William, Itziar Mendizabal, eta Idoia Santamaria. 2008. *Sistema eragileak: barnekoak eta diseinuko printzipioak*. Servicio Editorial Universidad del País Vasco-Euskal Herriko Unibertsitatea.

Stankovic, John A. 1988. Misconceptions about real-time computing: A serious problem for next-generation systems. *Computer* 21.10–19.

7. Eskerrak eta oharrak

Oharra: Lan honen edukia International Conference on Reliable Software Technologies (AEiC 2021) kongresuan aurkeztuko den ekarpenetik eratorria da. Izenburu originala *"How windows size and number can influence the schedulability of hierarchically-scheduled time-partitioned distributed real-time systems"* da. Egileak: Andoni Amurrio, Mario Aldea, J. Javier Gutiérrez eta Ekain Azketa.

Eskerrak eman nahi dizkiet nire zuzendariei. Batetik, Ikerlaneko Ekain Azketari, eta bestetik, Kantabriako Unibertsitateko Mario Aldea eta J. Javier Gutiérrezi. Udako Euskal Unibertsitateari ere nire esker ona adierazi nahi diot, IkerGazte bezalako komunitateak bultzatzeagatik eta, oro har, euskararen alde eginiko lanarengatik.