G.A. SHAHNAZARYAN

DEVELOPING THE BLOCK OF CORRELATION PREDICTION OF BRANCH INSTRUCTIONS IN MODERN PROCESSORS

A system, implementing the correlation prediction of branch instructions at pipelined processing in modern processors is considered, and the block diagram of the prediction block is developed. Estimates of the developed block of correlation prediction from the point of view of speed and hardware costs when designing based on FPGA are given.

Keywords: instruction-level parallelism (pipelining), correlation prediction system, branch history register (BHR), pattern history table (PHT).

ՀՏԴ 004.383.2

Լ.Յ. ԹՈՐՈՍՅԱՆ, Ծ.Ս. ՀՈՎՀԱՆՆԻՍՅԱՆ, Հ.Հ. ՇԻՏԻԿՅԱՆ

IoT ԵՎ ԿԼԻԵՆՏ-ՍԵՐՎԵՐ ՀԱՄԱԿԱՐԳԵՐԻ ԱՄՊԱՅԻՆ ՀԱՐԹԱԿՆԵՐՈՒՄ ՏԵՂԱԿԱՅՄԱՆ ԵՎ ԿԱՌԱՎԱՐՄԱՆ ՀԱՄԱԿԱՐԳԵՐԻ ՀԵՏԱԶՈՏՈՒԹՅՈՒՆ ԵՎ ՀԱՄԵՄԱՏԱԿԱՆ ՎԵՐԼՈՒԾՈՒԹՅՈՒՆ

Հետազոտվել են ամպային հարթակներում IoT և կլիենտ-սերվեր ճարտարապետությամբ համակարգերի տեղադրման եղանակները: Կատարվել է դրանց համեմատում: Դիտարկվել են կոնտեյներների միջոցով աշխատող համակարգերի օրկեստրացիայի գործիքամիջոցները, և կատարվել է դրանց համեմատական վերլուծություն:

Առանցքային բառեր. IoT, ամպային տեխնոլոգիաներ, կոնտեյներներ, կառավարում (օրկեստրացիա), Docker, Docker Swarm, Kubernetes:

Ներածություն: Կորպորատիվ հաճախորդներին սպասարկող և ծրագրային ապահովման՝ որպես ծառայության (Software-as-a-service, SaaS) լուծումները պահանջում են հաշվողական մեծ ռեսուրսներ: Նման լուծումներ առաջարկող կազմակերպությունները ժամանակ առ ժամանակ կարիք են ունենում ավելացնելու կամ նվազեցնելու իրենց համակարգերի աշխատանքն ապահովող ռեսուրսների քանակը [1,2]: Ավանդաբար նշված համակարգերը տեղակայվել են վիրտուալ մեքենաներով աշխատող սերվերային համակարգերում: Ամպային համակարգերի ի հայտ գալուց հետո համակարգերն աստիճանաբար տեղակայվում են դրանցում, ինչը թույլ է տալիս կատարել ֆինանսական զգալի տնտեսումներ, ինչպես նաև շատ արագ արձագանքել ծրագրային համակարգի պահանջարկի փոփոխությանը՝ ավելացնելով կամ նվազեցնելով կիրառվող ապարատային և ծրագրային ռեսուրսները:

Նմանատիպ միտում է նկատվում իրերի ինտերնտի (Internet-of-Things, IoT) ոլորտում [3]: Իրերի ինտերնետը ներառում է այնպիտի ոլորտներ, ինչպիսիք են խելացի տները, խելացի մեքենաները, խելացի քաղաքները և այլն: Ավանդական սերվերային կայանները ոչ միշտ է, որ կարող են արագ արձագանքել իրերի ինտերնետով աշխատող համակարգերի պահանջներին: Արդյունքում՝ ստեղծվել են մառախուղային և եզրային հաշվրակների համակարգեր (Fog Computing, Edge Computing), որոնց նպատակը IoT համակարգերին ամպային հարթակներին միացնելն է: Նման դինամիկ միջավայրում ամպային հարթակներում ռեսուրսների կառավարումը դառնում է առաջնային: Ավանդական վիրտուալ մեքենաներով աշխատող համակարգերը փոխարինվում են կոնտեյներային համակարգերով, ինչպիսին է կոնտեյացման Docker համակարգը: Կոնտեյներային համակարգերը կառավարելու համար կան մի քանի լուծումներ, որոնցից են Docker Swarm և Kubernates համակարգերը:

Խնդրի դրվածքը: Ամպային համակարգերում կլիենտ-սերվեր ճարտարապետությամբ և IoT համակարգերի տեղակայման և դրանց օգտագործվող ռեսուրսների կառավարման համար կիրառվում են համակարգեր, ինչպիսիք են Docker Swarm-ը և Kubernates-ը: Նման համակարգերն ունեն զգալի տարբերություններ, և կան խնդիրներ, որոնց լուծումն առավել լավ է կազմակերպված դրանցից մեկում: Արդիական է հետազոտել այս համակարգերը, կատարել դրանց համեմատություն և որոշել, թե խնդիրների որ խմբի համար դրանցից որն է առավել արդյունավետ:

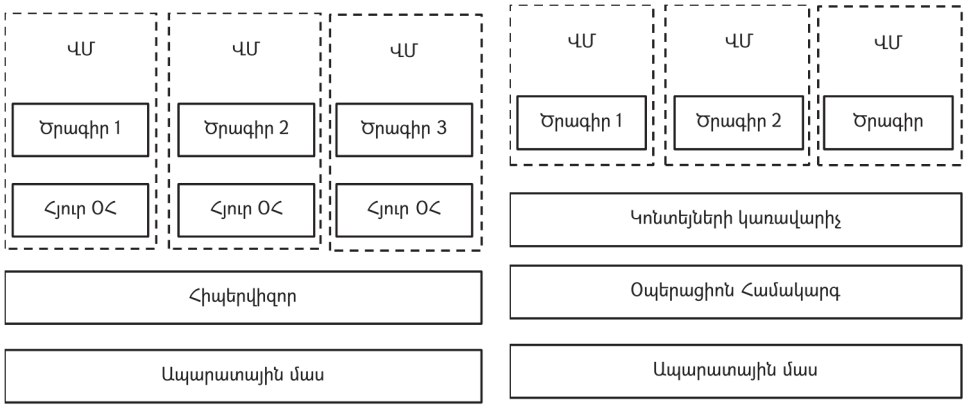
Վիրտուալ մեքենաներ և կոնտեյներացում: Վիրտուալ մեքենան (ՎՄ) քոմփյութերային համակարգի էմուլյացիա է: Դա թույլ է տալիս մեկ ֆիզիկական մեքենայի վրա միաժամանակ աշխատեցնել մի քանի օպերացիոն համակարգեր (ՕՀ) այնպես, ինչպես դրանք կաշխատեին, եթե տեղադրված լինեին առանձին ֆիզիկական մեքենաների վրա: ՎՄ-ների ՕՀ-երը կիսում են այն մեքենայի հաշվողական ռեսուրսները, որի վրա տեղակայված են: Ամեն ՎՄ ունի իր ՕՀ-ն, իսկ ապարատային ռեսուրսները վիրտուալացված են: ՎՄ-ների ստեղծման և դրանց ապարատային ռեսուրսների տրամադրման համար պատասխանատու է հիպերվիզորը: Հիպերվիզորը տեղակայված է ապարատային մասի և ՎՄ-ի արանքում [4]: Վերջինիս հիմանական նպատակը սերվերի վիրտուալացման ապահովումն է: ՎՄ-ի աշխատանքը ներկայացված է նկ.1 ա-ում:

ՎՄ-ները ծախսում են մեծ քանակի հաշվողական ռեսուրս, քանի որ պարունակում են ամբողջական ՕՀ և ապարատային այն մասի վիրտուալ պատճենը, որի վրա աշխատելու է: Նման ճարտարապետությունը նույնիսկ հասարակ ծրագրային համակարգի աշխատանքի համար ծախսում է զգալի քանակով հիշողու-

թյուն և պրոցեսորային ժամանակ, սակայն տնտեսապես առավել նպատակահարմար է:

ՕՀ-ների վիրտուալացումը լայն տարածում է ստացել և թույլ է տալիս ծրագրային համակարգերը հեշտությամբ մի սերվերային համակարգից տեղափոխել մեկ այլի: Սակայն հաշվողական ռեսուրսների զգալի օգտագործումը խնդիրներ է առաջացնում ժամանակակից միկրոսերվիսային ճարտարապետություններում:

Որպես նշված խնդրի լուծում կիրառվում են կոնտեյներները: Կոնտեյները տեղակայված է ֆիզիկական սերվերի և դրա ՕՀ-ի վրա: Յուրաքանչյուր կոնտեյներ կիսում է ՕՀ-ի կեռնելը [4]: Կոնտեյներները փոքր են, սովորաբար մի քանի հարյուր մեգաբայթից քիչ և գործարկվում են հաշված վայրկյանների ընթացքում: Կոնտեյներները նաև նվազեցնում են սերվերների սպասարկման ժամանակը, քանի որ սպասարկվում է միայն մեկ ՕՀ, որը համատեղ օգտագործում են բոլոր կոնտեյներները: Կոնտեյների կառուցվածքը ներկայացված է նկ.1 բ-ում:



ա) *բ)*

Նկ. 1. Սերվերային համակարգերի կառուցվածքները՝ ա) ՎՄ, բ) կոնտեյներ

Տարածված է կոնտեյներների երկու տեսակ.

1. Linux Containers (LXC) - Linux ՕՀ-ի կոնտեյներների տեխնոլոգիան է: Այն ՕՀ մակարդակի վիրտուալացումն է, որը թույլ է տալիս մեկ ՕՀ-ում աշխատեցնել մի քանի մեկուսացված Linux ՕՀ-ներ:

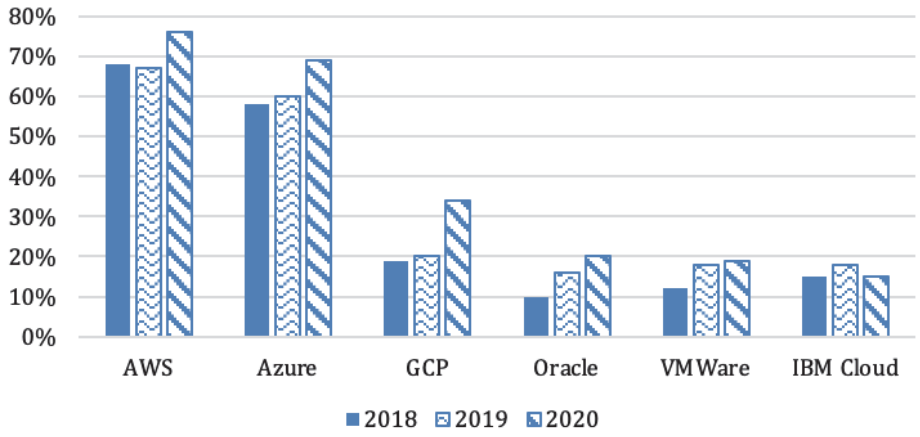
2. Docker - ստեղծվել է LXC-ի հիման վրա, սակայն ժամանակի ընթացքում կատարելագործվել է և այժմ հասանելի է նաև Windows ՕՀ-ներում:

3. Կոնտեյներացումը սկսում է մեծ կիրառություն ունենալ նաև IoT համակարգերում: Հետազոտությունները ցույց են տալիս, որ 2020 թվականից ավելի քան 20 միլիարդ IoT սարքեր են միացված լինելու միմյանց, որը կրկնակի ավել է, քան համակարգիչները, սմարթֆոնները և պլանշետներն իրար հետ վերցրած:

Կոնտեյներները թույլ են տալիս հեշտությամբ տեղակայել, թեստավորել և թարմացնել IoT ծրագրերը: Որպես կանոն, IoT համակարգերը սկզբնական փուլում մշակվում են SBCs (single-board computers) հարթակների վրա, որից է Raspberry Pi-ն: Վերջնական փուլում համակարգերը կարող են տեղակայվել SoC (system-on-a-chip) սարքերի վրա: SBCs-ի և SoC-ի վրա աշխատելիս կիրառվում են տարբեր հրամանային համակարգեր, սակայն կոնտեյներները թույլ են տալիս ծրագրային համակարգի հիմքի վրա ստեղծել տարբեր պատկերներ, որոնք կաշխատեն համապատասխան հրամանային համակարգերով, սակայն կունենան նույն ծրագրային կոդը: Դա զգալիորեն կրճատում է համակարգի մշակման ժամանակը և, հետևաբար, ինքնարժեքը:

Կոնտեյներների տարածվածության աճի հետ առաջանում է ամպային հարթակներում տեղադրման, կարգաբերման և կառավարման անհրաժեշտություն: «Գարթներ» ընկերության վերլուծաբանների կանխատեսման համաձայն՝ 2023 թվականի ավարտին կազմակերպությունների 70%-ից ավելին ունենալու են երեք կամ ավելի կոնտեյներ կիրառվող համակարգեր [5]:

«Ֆլեքսերա» ընկերության հետազոտությունները ցույց են տալիս, որ 1000-ից ավելի աշխատող ունեցող ընկերություններում բաց ամպային լուծումներից (public cloud) առավել տարածված են Amazon, Microsoft Google ընկերությունների համապատասխանաբար AWS, Azure և GCP ծառայությունները, որոնց հաջորդում են Oracle, VMWare և IBM ընկերության լուծումները (նկ.2):



Նկ. 2. Ամպային ծառայությունների տարածվածությունը

Կոնտեյներային համակարգերի կառավարման (օրկեստրացիայի) համակարգեր:

Կոնտեյներների տարածվածության նման պայմաններում անհրաժեշտություն է առաջանում արդյունավետ կառավարել դրանք: Ներկայումս շուկայում կա երկու ակտիվ դերակատար՝ Google ընկերությունն իր Kubernetes լուծումով և Docker ընկերությունը՝ Docker Swarm լուծումով [6,7]: Կոնտեյների կառավարումն այլ կերպ կոչվում է օրկեստրացիա: Կոնտեյների օրկեստրացիան առանձին կոնտեյներների ավտոմատ կառավարումն է:

Օրկեստրացիան ծրագրային համակարգերի համար տալիս է հետևյալ լուծումները.

• բազմաթիվ մեքենաներում ծրագրային համակարգի բաշխման կենտրոնացված գործիքամիջոց,

• հանգույցներից մեկի խափանման դեպքում նորի ստեղծում,

• ռեսուրսների արդյունավետ կառավարում,

• կարգավորումների ճկուն համակարգ:

Նշված լուծումների համեմատական վերլուծությունը բերված է աղ. 1-ում: Աղ. 2-ում բերված են դիտարկված լուծումների առավելություններն ու թերությունները:

Աղյուսակ 1

Kubernetes-ի և Docker Swarm-ի համեմատություն

	Kubernetes	Docker Swarm
Տեղակայում	բարդ	պարզ
Ընդլայնում	տեղի է ունենում դանդաղ	տեղի է ունենում արագ
Հասանելիություն	բարձր հասանելիություն	բարձր հասանելիություն
Ցանց	ունի ներդրված գործիքակազմ	ունի ներդրված գործիքակազմ
Հանգույցներ	մինչև 5000	մինչև 2000
Մոնիտորինգ	կա ներդրված լավ գործիքակազմ	ներդված գործիքակազմ չկա, անհրաժեշտ է տեղադրել երրորդ ընկերության լուծում:
Ծանրաբեռնվածության բաշխում	կատարվում է ձեռքով կամ լրացուցիչ համակարգերի տեղակայմամբ	կատարվում է ավտոմատ կերպով
Ինտերֆեյս	ունի ներկառուցված կառավարման վահանակ	չունի ներկառուցված լուծում

Աղյուսակ 2

Kubernetes-ի և Docker Swarm-ի առավելություններն ու թերությունները

	Kubernetes	Docker Swarm
Առավելություններ	Բաց կոդ Մոդուլյար Հարմար է մեծ համակարգերի կառավարման համար	Պարզ է կիրառումը Պարզ և արագ տեղադրում Բաց կոդ Հարմար է փոքր համակարգերի կառավարման համար
Թերություններ	Բարդ է կիրառումը Համատեղելի չէ Docker CLI և Compose գործիքների հետ	Սահմանափակ հնարավորություններ Կայուն չէ խափանումների նկատմամբ

Եզրակացություն: Բազմաթիվ են ծրագրային համակարգեր, որոնք աշխատում են կլիենտ-սերվեր ճարտարապետությամբ կամ IoT սարքավորումների հետ տեղակայված են ամպային հարթակներում: Մեծ կորպորատիվ հաճախորդներին սպասարկող համակարգերը կարիք ունեն դիմանիկ կերպով փոփոխելու ապարատային ռեսուրսները՝ ավելացնելով կամ նվազեցնելով սերվերների քանակը: Ամպային հարթակներում կոնտեյներացման տեխնոլոգիայի լայն տարածում ստանալուց հետո ռեսուրսների դինամիկ կառավարման համար լայնորեն կիրառվում են օրկեստրացիայի գործիքամիջոցները: Դրանցից առավել տարածված են Docker Swarm-ը և Kubernetes-ը: Չնայած երկու համակարգերն էլ ունեն նման գործառույթ, սակայն բավականին տարբերվում են միմյանցից:

Docker Swarm-ն առավել նպատակահարմար է կիրառել ոչ մեծ համակարգերի համար և այն դեպքերում, երբ դրա կարգաբերմամբ զբաղվող ինժեները չունի մեծ փորձ: Այն բավական պարզ է կիրառման առումով, համատեղելի է Docker CLI և Compose գործիքների հետ: Միևնույն ժամանակ, մեծ համակարգերի դեպքում այն այդքան էլ կիրառելի չէ, քանի որ ունի մի շարք սահմանափակումներ, և դրա միջոցով տեղակայված համակարգերի խափանումները դժվարությամբ են կառավարվում:

Kubernetes-ը հզոր և, որպես հետևանք, բարդ համակարգ է: Այն առավել արդյունավետ է կիրառել մեծ համակարգերի դեպքում, որտեղ առաջնային են արագ ընդլայնելիությունն ու համակարգի հասանելիությունը: Kubernetes-ը համատեղելի է բոլոր ամպային հարթակների հետ ու ունի կառավարման հզոր գործիքակազմ: Սակայն դրա տեղակայումն ու սպասարկումը պահանջում են որոշակի գիտելիքներ, և որոշ դեպքերում ժամանակատար է:

ԳՐԱԿԱՆՈՒԹՅԱՆ ՑԱՆԿ

1. Data Storage Security in Cloud Computing Using Container Clustering / **R. G.Rohan, M. Gaurav, K. Subham, et al** -August 2016.
2. https://www.microsoft.com/en-us/research/wp-content/uploads/2009/01/p68-v39n1o-greenberg.pdf
3. Internet of things: a survey on enabling technologies protocols and applications /**A. Al-Fuqaha, M. Guizani, M. Mohammadi, et al** // IEEE Communications Surveys Tutorials. – 2015. – 17.- P. 2347-2376.
4. https://docs.microsoft.com/en-us/virtualization/windowscontainers/about/containers-vs-vm
5. https://www.itopstimes.com/contain/enterprise-container-strategy-its-time-to-jump-on-board/
6. https://kubernetes.io/
7. https://docs.docker.com/engine/swarm/

Л.Ю. ТОРОСЯН, Ц.С. ОГАННЕСЯН, О.А. ШИТИКЯН

ИССЛЕДОВАНИЕ И СРАВНИТЕЛЬНЫЙ АНАЛИЗ СИСТЕМ РАЗВЕРТЫВАНИЯ И ОРКЕСТРАЦИИ IOT И КЛИЕНТ-СЕРВЕР СИСТЕМ В ОБЛАЧНЫХ ПЛАТФОРМАХ

Исследованы варианты установки IoT и клиент-серверной архитектуры на облачные платформы. Проведено их сравнение. Рассмотрены работающие с помощью контейнеров инструменты оркестровки систем и проведен их сравнительный анализ.

Ключевые слова: IoT, облачные технологии, контейнеры, оркестрация, Docker, Docker Swarm, Kubernete.

L.Y. TOROSYAN, Ts.S. HOVHANNISYAN, H.H. SHITIKYAN

INVESTIGATION AND COMPARATIVE ANALYSIS OF THE DEPLOYMENT AND ORCHESTRATION SYSTEMS IOT AND CLIENT-SERVER SYSTEMS IN CLOUD PLATFORMS

The variants of deployment of IoT and client-server architecture-based systems on cloud platforms are investigated. Those systems are compared. The orchestration tools of the systems working with containers are considered, and the comparative analysis is accomplished.

Keywords: IoT, cloud technologies, containers, orchestration, Docker, Docker Swarm, Kubernetes.