UO‘K 004.622

KATTA MA’LUMOTLAR OQIMINI QAYTA ISHLASH: KECHIKISHLARNI OPTIMALLASHTIRISH VA RESURSLARNI DINAMIK BOSHQARISH ALGORITMLARI TAHLILI

[+] *Boynazarov I.M.*[1], *Mamanov J.*[2], *Mahmudov J.I.*[1], *Esonboyev M.*[1],

[1] Muhammad al-Xorazmiy nomidagi Toshkent axborot texnologiyalari universiteti Samarqand filiali, Samarqand, O‘zbekiston
[1] Denov tadbirkorlik va pedagogika instituti, Denov, O‘zbekiston

[+] ilhomboy1974@gmail.com

Annotatsiya. Ushbu tahliliy maqolada katta ma’lumotlar oqimini real vaqtda qayta ishlash tizimlarida kechikishlarni kamaytirish va resurslarni dinamik boshqarishga qaratilgan zamonaviy arxitektura, algoritm va yondashuvlar tizimli ravishda ko‘rib chiqildi. Maqolada Lambda, Kappa va gibrid arxitekturalarning xususiyatlari, cloud-native platformalar, shuningdek, Edge-Fog-Cloud gibrid muhitlari tahlil qilindi. Tarmoq va hisoblash darajasidagi kechikish omillari, ularni kamaytiruvchi usullar – operator joylashishi, vazifalarni uzatish (task offloading), RDMA texnologiyasi va mashina o‘rganishga asoslangan bashorat modellari o‘rganildi. Bulutli muhitda resurslarni dinamik boshqarish masalalari, shu jumladan reaktiv, proaktiv va gibrid avto-masshtablash algoritmlari, vaqt qatorlari asosidagi, chuqur o‘rganish va mustahkamlovchi o‘rganishga tayanuvchi yondashuvlar, ko‘p mezonli vazifalar rejalashtirilishi tahlil qilindi. Tahlil natijalari shuni ko‘rsatadiki, eng yuqori samaradorlikka turli yondashuvlarni – proaktiv bashorat, gibrid avto-masshtablash va ML asosidagi qaror qabul qilishni – integratsiyalash orqali erishish mumkin. Hozirda hal etilmagan muammolar va istiqbolli tadqiqot yo‘nalishlari aniqlab olindi.

Kalit so‘zlar: katta ma’lumotlar, oqimli qayta ishlash, Lambda arxitekturasi, Kappa arxitekturasi, bulutli hisoblash, kechikish, operator joylashishi, avto-masshtablash, mashina o‘rganish, resurslarni dinamik boshqarish, Edge-Fog-Cloud.

1 KIRISH

Zamonaviy raqamli iqtisodiyotda ma'lumotlar hajmining misli ko‘rilmagan darajada o‘sishi – IoT qurilmalari, telekommunikatsiya tarmoqlari, sanoat sensori tizimlari va multimedia xizmatlari kabi manbalardan sekundiga millionlab hodisalar oqimining generatsiya qilinishi – an’anaviy batch-yo‘naltirilgan qayta ishlash modellarining imkoniyat chegarasidan oshib ketishi bilan xarakterlanadi [1, 2]. Dunyo aholisi 2050-yilga kelib 9.7 milliardga yetishi kutilayotgan bir sharoitda raqamli tizimlar ham murakkabligi va hajmi jihatidan misli ko‘rilmagan suratlarda o‘smoqda [8]. Bunday sharoitda real vaqt rejimida ishlovchi oqimli qayta ishlash tizimlari muhim ahamiyat kasb etib, ular yuqori o‘tkazuvchanlik, minimal kechikish va taqsimlanish imkoniyatlarini bir vaqtda ta’minlashi zarur [3].

Ushbu talablarni bajarish tizim arxitekturasi, resurslarni joylashtirish strategiyasi va tarmoq infratuzilmasi bilan chambarchas bog‘liq [9]. Lambda va Kappa arxitekturalari, shuningdek, cloud-native va Edge-Cloud gibrid platformalari bugungi kunda oqimli qayta ishlash tizimlarining asosiy arxitekturaviy modellariga aylangan [3, 4, 6]. Bir vaqtning o‘zida kechikishni kamaytirish va resurslarni samarali boshqarish masalalari tadqiqotchilar e'tiborini tobora ko‘proq jalb etmoqda [19, 21]. Cloud hisoblashning asosiy imtiyozlaridan biri – resurslarning elastikligi bo‘lib, bu foydalanuvchilarga talabga mos ravishda infrastruktura hajmini oshirish yoki kamaytirish imkonini beradi [2].

Bulutli muhitda resurslarni statik taqsimlash yondashuvlari yuklama o‘zgarishlariga moslasha olmasligi sababli ko‘pincha resurs yetishmovchiligi yoki ortiqcha ajratilish holatlariga olib kelmoqda [2, 23]. Kuzatishlar shuni ko‘rsatadiki, ma’lumotlar markazlarida serverlarning o‘rtacha ish yuklanish koeffitsienti 5 foizdan 20 foizgacha bo‘lib, qolgan vaqt resurslar bekor turadi [2]. Shu sababli avto-masshtablash mexanizmlarini qo‘llash zarurati yuzaga kelmoqda. Reaktiv yondashuvlar tizimning joriy holatiga asoslanib qaror qabul qilsa, proaktiv yondashuvlar kelajakdagi yuklamani oldindan bashorat qilib, resurslarni oldindan ajratadi [3, 21].

Mashina o‘rganishi va chuqur o‘rganish usullari ushbu jarayonlarni yanada samaraliroq tarzda avtomatlashtirishga imkon bermoqda [25, 32]. Xususan, chuqur mustahkamlovchi o‘rganish (Deep Reinforcement Learning – DRL) algoritmlari murakkab, dinamik muhitlarda qaror qabul qilish uchun katta imkoniyatlar ochmoqda va bulutli hisoblashda vazifalarni rejalashtirilish va resurslarni boshqarish uchun eng istiqbolli yo‘nalishlardan biriga aylanmoqda [32]. Shu bilan birga, ko‘p mezonli optimallashtirish yondashuvlari bajarilish vaqti, energiya sarfi, kechikish va iqtisodiy xarajatlarni birgalikda muvozanatlashtirish imkonini beradi [24, 25].

Ushbu maqolaning maqsadi – katta ma’lumotlar oqimini real vaqtda qayta ishlash tizimlarida kechikishni optimallashtirish va resurslarni dinamik boshqarishga doir mavjud ilmiy tadqiqotlarni tizimli ravishda tahlil qilish, ularning afzalliklari va cheklovlarini aniqlash hamda istiqbolli tadqiqot yo‘nalishlarini belgilab olishdan iborat. Maqola ikkita o‘zaro bog‘liq ilmiy yondoshuv – oqimli qayta ishlash tizimlarining arxitekturalari va kechikishni optimallashtirish hamda resurslarni dinamik boshqarishni birgalikda tahlil qilishga asoslangan.

2 KATTA MA’LUMOTLAR OQIMINI QAYTA ISHLASH TIZIMLARI ARXITEKTURALARI

Katta ma’lumotlar oqimini qayta ishlash tizimlarining samaradorligi ko‘p jihatdan tanlangan arxitekturaviy modelga bog‘liq. Zamonaviy platformalar ma’lumotlarni qabul qilish, qayta ishlash va saqlash jarayonlarini qatlamli tuzilmalar orqali tashkil etadi, bu esa taqsimlanish va moslashuvchanlikni ta’minlaydi [3, 4]. Lambda va Kappa arxitekturalari, cloud-native platformalar hamda gibrid Edge-Fog-Cloud muhitlari bugungi kunda eng keng qo‘llaniladigan yondashuvlar hisoblanadi. Har bir yondashuv ma'lum qo‘llanilish holatlari, imkoniyatlari va cheklovlariga ega.

Katta ma’lumotlar – hajm (volume), tezlik (velocity), xilma-xillik (variety), ishonchlilik (veracity) va qiymat (value) kabi 5V xususiyatlari bilan tavsiflanadi [8]. Ushbu xususiyatlar an’anaviy ma’lumotlar bazasi tizimlari va qayta ishlash vositalarining imkoniyat chegarasidan oshib ketadi, bu esa yangi arxitekturaviy yechimlarni talab etadi. Shu maqsadda Lambda, Kappa va gibrid arxitekturalar paydo bo‘lgan.

2.1 Lambda arxitekturasi

Lambda arxitekturasi – ko‘p hajmli ma’lumotlarni batch va real vaqt qayta ishlashni birlashtirish orqali boshqarishga mo‘ljallangan gibrid model bo‘lib, u uchta asosiy qatlamdan iborat: batch qatlami, tezlik qatlami va xizmat ko‘rsatish qatlami [8, 34]. Batch qatlamda katta hajmli tarixiy ma’lumotlar Apache Hadoop yoki Spark kabi vositalarda qayta ishlanib, to‘liq va aniq ko‘rinish hosil qilinadi. Batch qatlam ma’lumotlarning doimiy saqlanish joyi (master dataset) vazifasini bajaradi va bu ma’lumotlar o‘zgarmas (immutable) ko‘rinishda saqlanadi.

Tezlik qatlamida esa real vaqt ma’lumotlari Apache Storm, Apache Flink yoki Spark Streaming yordamida qayta ishlanib, joriy hodisalarga tezkor javob beriladi [8]. Batch qatlami katta hajmli ma’lumotlarni qayta ishlashda juda ko‘p vaqt talab qilganligi sababli natijalar ko‘pincha yangilanmagan bo‘ladi. Tezlik qatlami ushbu bo‘shliqni eng so‘nggi ma’lumotlarni so‘rovlar asosida ta’minlash orqali bartaraf etadi. Nihoyat, xizmat ko‘rsatish qatlami (serving layer) yuqoridagi ikkala qatlamning natijalarini birlashtiradi.

Lambda arxitekturasining asosiy afzalligi – uning yuqori ishonchlilik va nosozliklarga bardoshlilikdir. Batch qatlam orqali ma’lumotlarning to‘liq tarixiy ko‘rinishi saqlanib, tezlik qatlami orqali esa eng so‘nggi ma’lumotlar asosida tezkor natijalar taqdim etiladi [11]. Ushbu ikki qatlamli yondashuv xatoliklarga nisbatan yuqori bardoshlilikni ta’minlaydi: agar tezlik qatlamida xato yuz bersa, batch qatlamning to‘liq va aniq ma’lumotlari asosida tizim tiklana oladi.

Biroq ushbu arxitekturaning muhim kamchiligi shundaki, batch va streaming komponentlari uchun bir xil biznes-mantiqni ikki alohida tizimda amalga oshirish talab etiladi. Bu esa tizim murakkabligini oshiradi va dasturiy ta’minot xarajatlarini ko‘paytiradi [3]. Bundan tashqari, ikkala tizim uchun alohida kod bazasini saqlash, yangilash va sinab ko‘rish operatsion yukni oshiradi. Lambda arxitekturasi yirik telekommunikatsiya kompaniyalari, moliya institutlari va aqlli shahar infratuzilmalarida keng qo‘llaniladi [13]. ADS-B ma’lumotlari asosida havo-yer kuzatuv tizimlarida qo‘llangan tadqiqot Lambda arxitekturasining turli og‘ir stsenariylarda 100% aniqlik darajasini ta’minlashini isbotlagan [34].

2.2 Kappa arxitekturasi

Kappa arxitekturasi Lambda arxitekturasining cheklovlarini bartaraf etish maqsadida Apache Kafka hammuassisi Jay Kreps tomonidan 2014-yilda taklif etilgan bo‘lib, u faqat bitta stream processing tizimidan foydalanadi [4]. Kappa arxitekturasida barcha ma’lumotlar – tarixiy yoki real vaqt – oqim sifatida ko‘rib

chiqiladi va ularning hammasi bitta stream processing mexanizmi orqali qayta ishlanadi [12]. Ushbu model Apache Kafka kabi o‘zgarmas jurnal (immutable log) atrofida quriladi.

Kappa arxitekturasida kiruvchi ma’lumot hodisalari markaziy o‘zgarmas jurnal – masalan, Kafka mavzusi – ga uzluksiz ravishda qo‘shib boriladi. Yagona stream processing mexanizmi (Apache Flink, Kafka Streams yoki Spark Structured Streaming) ushbu jurnalni o‘qib, real vaqt ko‘rinishlarini yoki kelib chiqartirilgan ma’lumotlar omborini yangilab boradi [4, 8]. Tarixiy ma’lumotlarni qayta ishlash zarur bo‘lganda, bir xil streaming mexanizmi tomonidan ma’lumotlar oqimi boshidan qayta ishlatiladi – bu batch qatlam talab etmasdan tarixiy tahlilni amalga oshirishga imkon beradi.

Kappa arxitekturasi IoT yo‘naltirilgan muhitlarda, jumladan aqlli qishloq xo‘jaligi tizimlarida muvaffaqiyatli qo‘llanilgan [4, 8]. Ushbu arxitekturaning asosiy ustunligi – tizim murakkabligini pasaytirishi va texnik xizmat ko‘rsatishni soddalashtirishi. Bitta kod bazasi va bitta qayta ishlash infratuzilmasini saqlab qolish ham ishlab chiqish xarajatlarini kamaytiradi. Yuqori o‘tkazuvchanlikka ehtiyoj va past kechikish talablari bo‘lgan ilovalarda Kappa arxitekturasi mos keladi.

Biroq Kappa arxitekturasi ham o‘ziga xos cheklovlarga ega. Tarixiy ma’lumotlarni qayta ishlashda zarur bo‘lganda, ma’lumotlar oqimini boshidan boshlab qayta ishlatish hisoblash resurslariga qo‘shimcha yuk tushirishi mumkin [34]. Bundan tashqari, barcha ma’lumotlarni oqim sifatida ko‘rib chiqish – jarayon va tarixiy ma’lumotlarning algoritmlar o‘zaro farq qiladigan murakkab holatlarda – ba’zan samarasiz bo‘lishi mumkin.

2.3 Cloud-Native oqimli qayta ishlash platformalari

Cloud-native arxitekturalar mikroservislar, konteynerizatsiya (Docker, Kubernetes) va orkestratsiya texnologiyalariga asoslanadi [6, 7]. Bunday muhitlarda stream processing pipeline-ning har bir funksional komponenti mustaqil xizmat sifatida ishlatiladi va alohida miqyoslanishi mumkin [3]. Shu sababli ma’lumotlarni qabul qilish, qayta ishlash, saqlash va tahlil qilish qatlamlarini bir-biridan ajratish moslashuvchanlikni oshiradi va alohida komponentlarni dinamik miqyoslash imkonini beradi.

Telekommunikatsiya ilovalarida cloud-native platformalar yuqori trafik sharoitida real vaqt tahlilotlari, monitoring va qaror qabul qilish vazifalarini samarali bajarishda kuchli qo‘llab-quvvatlashni ko‘rsatadi [13]. PROMENADE platformasi – aqlli shaharlar infratuzilmasini dinamik graflar orqali modellashtirib, Lambda va Kappa arxitekturalarini qo‘llagan holda real vaqt qayta ishlashni ta’minlovchi ochiq manbali cloud-native yechim – ushbu yondashuvning amaliy namunasi hisoblanadi [6]. Platforma Apache Kafka, MongoDB, Neo4J va Apache Spark kabi texnologiyalar to‘plamiga asoslanadi. OpenShift/Kubernetes muhitida joylashtirilgan platforma Lyon shahridagi yo‘l trafigi ma’lumotlarini (317,000 dan ortiq yo‘l segmenti) real vaqtda qayta ishlab, o‘rtacha 25 daqiqa kechikish bilan betweenness centrality ko‘rsatkichlarini hisoblab beradi [6].

WALLeSMART platformasi – Belgium Walloniya mintaqasidagi sutli fermer xo‘jaliklarini boshqarish uchun ishlab chiqilgan cloud-native yechim – Lambda arxitekturasiga asoslanib, Apache Kafka, Apache Storm va Hadoop vositalarini birlashtirib real vaqt va batch qayta ishlash imkoniyatlarini ta’minlaydi [8]. 30 ta fermer xo‘jaligi va 45 ta agrometerologik stansiyadan yig‘ilgan ma’lumotlar bilan sinovdan o‘tkazilgan ushbu platforma real vaqt ishlarini o‘rtacha 1 daqiqa 18 soniyada yakunlashi isbotlangan. Ma’lumotlar platformalari uchun jarayon-yo‘naltirilgan loyihalash metodologiyasi [7] AWS, Azure va Google Cloud ekotizimlarida optimal xizmatlar to‘plamini aniqlash uchun chiziqli dasturlash optimallashtirishiga asoslangan.

2.4 Gibrid Edge-Fog-Cloud arxitekturalari

Sof cloud asosidagi qayta ishlash vaqtga sezgir ilovalar – sanoat avtomatikasi, o‘z-o‘zini boshqaruvchi transport, real vaqt tibbiy monitoring – uchun nomaqbul kechikishni keltirib chiqarishi mumkin. Shu sababli Edge va Fog hisoblashni o‘z ichiga olgan gibrid arxitekturalar taklif etilgan [11, 14, 19]. Uch darajali Edge-Fog-Cloud modeli ma’lumot manbalari yaqinida dastlabki qayta ishlash (Edge), mintaqaviy aggregatsiya va koordinatsiya (Fog) va markazlashtirilgan murakkab tahlil (Cloud) funksiyalarini ajratib belgilaydi.

Edge-Cloud arxitekturasida dastlabki ma’lumotlarni filtrlash va agregatsiya ma’lumot manbalari yaqinida amalga oshiriladi, murakkab hisob-kitoblar esa cloud ma’lumotlar markazlarida bajariladi [19]. Bu yondashuv tarmoq trafigini kamaytiradi va javob vaqtini yaxshilaydi. Katta hajmli hisoblash vazifalari uchun moslashuvchan gibrid Edge-Cloud hamkorlik ofloading usuli [19] tarmoq kengligi va kechikish cheklovlarini hisobga olib, vazifalarni dinamik ravishda taqsimlash imkonini beradi.

PHWAN – Industry 4.0 aqlli ishlab chiqarish uchun mo‘ljallangan gibrid platforma bo‘lib, Edge-Cloud hamkorligini, past kechikishli tarmoq mexanizmlarini va xavfsizlik tizimlarini integratsiyalashgan holda real vaqt sanoat ilovalarini qo‘llab-quvvatlaydi [14]. Fog hisoblash Edge va Cloud o‘rtasidagi oraliq qatlam sifatida kechikishga sezgir tahlil va koordinatsiyani qo‘llab-quvvatlaydi. Aqlli energiya boshqaruv tizimida

Edge-Fog-Cloud gibrid freymvorki metasistematik optimallashtirish asosidagi bashorat bilan birgalikda qo‘llanganda energiya samaradorligini sezilarli oshirishi ko‘rsatilgan [20].

Ma’lumotlar platformalari uchun jarayon-yo‘naltirilgan loyihalash metodologiyasi [7] Lambda, Kappa va Lakehouse kabi arxitekturaviy andozalarni tanlab, optimal bulut xizmatlarini aniqlash uchun chiziqli dasturlash optimallashtirish muammosi sifatida formallashtiradi. Amalga oshirilgan ikkita real holdagi sinovlar – KPI Factory (Microsoft Azure) va Customer Lifetime Value bashorat (Google Cloud) – dasturiy ta’minotchilarsiz ham to‘g‘ri loyihalash erishish mumkinligini isbotlagan [7].

2.5 Arxitekturaviy modellarning qiyosiy tahlili

Lambda, Kappa va gibrid arxitekturalarni tanlash ko‘p jihatdan ilovaning o‘ziga xos talablariga bog‘liq. Lambda arxitekturasi tarixiy va real vaqt ma’lumotlarini bir vaqtda qayta ishlash kerak bo‘lgan holatlarda – masalan, moliyaviy tahlil, katta hajmli log-tahlil – afzal hisoblanadi. Kappa arxitekturasi faqat real vaqt ma’lumotlariga tayanadigan va sodda texnik tuzilishni afzal ko‘radigan ilovalar uchun – IoT monitoring, fraud detection, real vaqt tavsiyalar tizimlari – optimal tanlov. Gibrid yondashuvlar esa ikkala talabni bajarishni talab qiladigan murakkab sanoat tizimlarida qo‘llaniladi.

Texnologiyalar darajasida ham sezilarli farqlar mavjud. Lambda arxitekturasida Apache Hadoop yoki Spark batch qayta ishlash uchun, Apache Storm yoki Flink real vaqt uchun, Apache Kafka esa ikkala qatlam uchun xabar broker sifatida keng qo‘llaniladi [8]. Kappa arxitekturasida Apache Kafka markaziy immutable jurnal, Apache Flink yoki Kafka Streams esa yagona stream processing mexanizmi vazifasini bajaradi [4]. Cloud-native muhitlarda Kubernetes va Docker orkestratsiyasi barcha arxitekturalar bilan birga qo‘llaniladi, bu esa miqyoslanish va xatoliklarga bardoshlilikni ta’minlaydi [6].

Ushbu arxitekturaviy modellarning amaliy qo‘llanilishi turli sohalar uchun farqlanadi. Aqlli shaharlar kontekstida PROMENADE [6] platforma Lambda/Kappa tamoyillarini birlashtirib, grafga asoslangan shahar infratuzilmasi modelini real vaqtda yangilab boradi. Qishloq xo‘jaligi sohasida WALLeSMART [8] Lambda arxitekturasini sut fermalarini boshqarish uchun qo‘llab, batch va real vaqt qayta ishlashni birlashtirib samarali natijalar ko‘rsatadi. Telekommunikatsiyada [13] esa bulutli platformalar yuqori trafik ostida real vaqt qaror qabul qilish uchun keng qo‘llanilmoqda. Har bir soha o‘ziga xos kechikish talablari, ma’lumot hajmi va arxitektura cheklovlarini talab etadi.

Arxitektura tanlash jarayonida ma’lumotlar platformalari uchun jarayon-yo‘naltirilgan metodologiya [7] muhim amaliy vosita hisoblanadi. Ushbu metodologiya ma’lumot oqimi diagrammasi (DFD) asosida optimal bulut xizmatlarini tanlashni chiziqli dasturlash masalasi sifatida formallashtiradi. Shunday qilib, Lambda yoki Kappa arxitekturasi xususiyatlarini o‘z ichiga olgan arxitekturaviy andozalar (design patterns) avtomatik ravishda aniqlanadi va tegishli xizmatlar bilan bog‘lanadi. Ushbu yondashuv AWS va Microsoft Azure kabi turli ekotizimlarda sinab ko‘rilgan va ekspert tavsiyalariga yaqin natijalarga erishilgan [7].

3 OQIMLI QAYTA ISHLASH TIZIMLARIDA KECHIKISH VA O‘TKAZUVCHANLIK

Kechikish va o‘tkazuvchanlik oqimli qayta ishlash tizimlarida xizmat sifatini (QoS) belgilovchi eng muhim ko‘rsatkichlar hisoblanadi. Ko‘plab real vaqt ilovalari yuqori ma’lumotlar qabul qilish tezligini saqlab, juda past javob vaqtini talab etadi [9, 14]. Ushbu ikkita ko‘rsatkich o‘rtasidagi muvozanat murakkab muammo bo‘lib, zamonaviy tadqiqotlar ularni birgalikda optimallashtirishga qaratilgan [9]. Xizmat darajasi kelishuvi (SLA) buzilishi ham resurs boshqaruvi sifatini baholashda asosiy mezon hisoblanadi.

3.1 Kechikish va o‘tkazuvchanlik tushunchalari

Kechikish (latency) – ma’lumot elementi yaratilgan paytdan boshlab uning qayta ishlash natijasi mavjud bo‘lgunga qadar o‘tgan umumiy vaqtni anglatadi. Bu davr ma’lumotlarni uzatish vaqti, navbatdagi kutish vaqti va amaliy qayta ishlash vaqtini o‘z ichiga oladi. O‘tkazuvchanlik esa birlik vaqt ichida qayta ishlanadigan ma’lumotlar miqdorini bildiradi. Sanoat va telekommunikatsiya tarmoqlarida past kechikishli uzatish mexanizmlarini joriy etish tizimning umumiy javob berish qobiliyatini sezilarli yaxshilaydi [15, 16]. Yuqori o‘tkazuvchanlikli stream processing frameworklari sekundiga millionlab hodisalarni qabul qilish qobiliyatiga ega bo‘lishi lozim [9].

Kechikish va o‘tkazuvchanlik o‘rtasida o‘zaro bog‘liqlik mavjud: o‘tkazuvchanlikni oshirish ko‘pincha resurs bandligining ortishiga va shu orqali qayta ishlash kechikishining oshishiga olib keladi [9]. SLA talablarini bajarmaslik tizimning amaliy qiymatini sezilarli darajada pasaytiradi. Masalan, kechikish 40 soniyadan oshgan taqdirda foydalanuvchilar platformani tark etishi mumkin – bu hol Friendster ijtimoiy tarmog‘ining tanazzulida namoyon bo‘lgan [2]. Shu sababli kechikish va o‘tkazuvchanlikni bir vaqtda optimallash muhim ahamiyat kasb etadi.

MapReduce modelida sekinlovchi vazifalar muammosi ham kechikish nuqtai nazaridan muhim ahamiyat kasb etadi. So‘nggi bir nechta map yoki reduce vazifalarni odatdagidan ancha uzoq vaqt bajaradigan mashinalar butun operatsiyaning yakunlanish vaqtini oshirishning keng tarqalgan sabablaridan biri hisoblanadi [1, 4]. Zaxira vazifalarni (backup tasks) rejalashtirish mexanizmi bu muammoni amalda hal etib, sort operatsiyasining bajarilish vaqtini 44% ga qisqartirib bo‘lishini isbotlagan [1].

3.2 Tarmoq darajasidagi kechikish omillari

Tarmoq darajasidagi kechikish kengligi, marshrutlash strategiyalari, paket yo‘qotilishi va qayta uzatishlar bilan bog‘liq. 5G va undan keyingi avlod tarmoqlarida ko‘p ulanish (multi-connectivity) va paketlarni takrorlash (packet duplication) texnikalari ishonchlilikni oshirish va uchidan-uchiga kechikishni kamaytirish uchun qo‘llaniladi [16]. Dinamik paketlarni takrorlash algoritmining 5G NR-DC tarmoqlarida mobillik sharoitida ishonchliligi va past kechikishliligini ta’minlashda samarali ekanligi isbotlangan [16].

RDMA asosidagi ma’lumotlarni uzatish texnologiyalari yadroi darajasidagi ishlov berishni chetlab o‘tib, an’anaviy TCP/IP aloqaga nisbatan sezilarli darajada past kechikishni ta’minlaydi [17]. RDMA yadro tomonidan ishlov berishni chetlab o‘tish orqali tarmoq kechikishini mikrosekund darajasiga tushirish imkonini beradi. Bu texnologiyalar, ayniqsa, ma’lumot intensiv stream processing pipeline-lari uchun juda foydali. Sanoat 50G-PON tarmoqlarida katta ulanishlar uchun past kechikishni ta’minlovchi tarmoq samaradorligi baholash mezonlari modeli [15] ushbu sohadagi tadqiqotlarning muhim qismini tashkil etadi.

MapReduce klasterlarida ommabop ma’lumotlarga nisbatan yuqori mos kelish holatlari sezilarli muammo hisoblanadi. Microsoft Bing klasterining 6 oylik ishlab chiqarish jurnallari tahlili shuni ko‘rsatdiki, eng ommabop ma’lumotlarning 12 foizi eng past uchdan bir qismdagi ma’lumotlarga qaraganda 10 martadan ko‘proq kuzatiladi; 18 foiz ma’lumot kamida uch bir vaqtli kirish operatsiyasiga ega [4]. Ushbu hotspot muammosi mahsulot operatsiyalarida Dryad ishi bajarilish vaqtini median bo‘yicha 16,7% ga oshiradi. Scarlett tizimi ushbu muammoni ma’lumotlarning kirish naqshlari asosida fayllarni maqsadli replikatsiya qilish yo‘li bilan hal qilib, Hadoop ishlarini median bo‘yicha 20,2% ga tezlashtiradi [4].

3.3 Hisoblash darajasidagi kechikish omillari

Hisoblash darajasidagi kechikish asosan hisoblash resurslarining yetishmasligi, operatorlarni noto‘g‘ri joylashtirish va klaster tugunlari bo‘yicha ish yukini notekis taqsimlashdan kelib chiqadi. Tadqiqotlar shuni ko‘rsatadiki, stream processing operatorlarini optimal joylashtirish tugunlar orasidagi aloqa xarajatlarini va umumiy kechikishni sezilarli kamaytirishi mumkin [9]. Operator joylashishi masalasi yo‘naltirilgan asiklik graflar (DAG) asosida optimallash muammosi sifatida formallashtirilib, yechiladi.

Kechikishni hisobga olgan operator joylashishi standart rejalashtirish strategiyalariga nisbatan javob vaqtini sezilarli darajada kamaytirishi isbotlangan [9]. Moslashuvchan oqim boshqaruvi ham qayta ishlash parametrlarini joriy tizim holatiga qarab dinamik tarzda sozlash imkonini beradi – masalan, oqim tezligi, oyna o‘lchami va bufer siyosati – tirbandlik va navbat kechikishining oldini olish uchun [9].

Qo‘shimcha ravishda, Edge yoki Fog tugunlariga vazifalarni uzatish kechikishga sezgir hisob-kitoblarni ma’lumot manbalari yaqinida bajarishga imkon beradi [19]. Cloud gaming ilovalarida real vaqt kechikishni bashorat qilish uchun yangi usullar ishlab chiqilmoqda [31]. Petri Net asosidagi freymvork Edge Cloud Continuum muhitida resurslarni rejalashtirish siyosatlarini modellashtirish va simulyatsiya qilishda qo‘llanilmoqda [30]. Mashina o‘rganishiga asoslangan bashorat modellari kelajakdagi ish yukini taxmin qilish va resurslarni proaktiv tarzda ajratish, shu orqali navbatdagi kechikishlar va qayta ishlash kechikishlarini kamaytirishda tobora ko‘proq qo‘llanilmoqda [18, 22].

3.4 IoT va real vaqt ilovalarida kechikish talablari

Internet of Things (IoT) qurilmalar sonining misli ko‘rilmagan darajada o‘sishi – 2023-yilga kelib global miqyosda 12 millionga yaqin qishloq xo‘jaligi sensori o‘rnatilishi kutilgan [8] – oqimli qayta ishlash tizimlariga yangi kechikish talablarini qo‘ymoqda. Industrial IoT ilovalarida – to‘liq avtomatlashtirilgan ishlab chiqarish liniyalari, real vaqt tibbiy monitoring, o‘z-o‘zini boshqaruvchi transport – kechikish millisekundlar darajasida bo‘lishi zarur bo‘lsa, aqlli qishloq xo‘jaligi tizimlarida bir necha daqiqali kechikish ham qabul qilinishi mumkin [6, 8].

IoT qurilmalar tomonidan yaratilgan ma’lumotlar hajmi ham tobora oshib bormoqda. IBM ma’lumotlariga ko‘ra, o‘rtacha bir fermer xo‘jaligi kuniga yarim million ma’lumot nuqtasini generatsiya qiladi [8]. Bunday katta hajmdagi ma’lumotlarni bulutga uzatish va u yerda qayta ishlash ko‘pincha tarmoq bottleneck-lari va kechikish muammolarini keltirib chiqaradi. Shu sababli Edge va Fog hisoblash paradigmasi IoT ilovalarida ayniqsa muhim: dastlabki ma’lumotlarni qurilma yaqinida qayta ishlab, faqat zarur axborotni bulutga yuborish tarmoq yukini sezilarli kamaytiradi [14, 19].

PROMENADE platformasining yo‘l trafigi monitoring ilovasi [6] IoT ma’lumotlarini real vaqtda qayta ishlashning yaxshi namunasidir. GPS qurilmali avtomobillardan sekundiga katta hajmda kelayotgan ma’lumotlar Edge qatlamida agregatsiya qilinib (3 daqiqalik oraliqlar uchun o‘rtacha sayohat vaqti hisoblanadi), so‘ngra bulut platformasiga yuboriladi. Bu yondashuv tarmoq yukini sezilarli kamaytiradi va end-to-end kechikishni boshqariladigan chegarada ushlab turadi [6]. Shu kabi WALLeSMART platformasida [8] sensorlar ma’lumotlari MQTT broker orqali real vaqtda qabul qilinib, Apache Storm yordamida darhol qayta ishlanadi, natijalar esa PostgreSQL va Cassandra ma’lumotlar bazalarida saqlanadi.

4 BULUTLI MUHITDA RESURSLARNI DINAMIK BOSHQARISH

Bulutli va klaster hisoblash muhitlarida resurslarni boshqarish hisoblash quvvati, tezkor va doimiy xotira, saqlash tizimlari hamda tarmoq resurslaridan samarali va muvozanatli foydalanishni ta'minlashga qaratilgan murakkab boshqaruv jarayonlari majmuasidir [2, 23]. Zamonaviy ilovalar dinamik va notekis yuklama sharoitida ishlashi sababli resurslarni statik taqsimlash ko‘pincha past samaradorlikka, resurs tanqisligiga yoki ortiqcha ajratilish holatlariga olib keladi.

4.1 Resurslarni boshqarish asoslari va navbat nazariyasi

Resurslarni boshqarish tizimlarining asosiy maqsadi xizmat sifati (QoS) talablarini ta'minlagan holda hisoblash infratuzilmasidan maksimal darajada samarali foydalanishdir [3]. Bunda kechikish, o‘tkazuvchanlik, energiya sarfi hamda iqtisodiy xarajatlar kabi ko‘rsatkichlar asosiy baholash mezonlari sifatida qaraladi [24]. SLA (Service Level Agreement) esa xizmat ko‘rsatuvchi va foydalanuvchi o‘rtasidagi kelishuv bo‘lib, minimal QoS talablarini belgilaydi.

Navbat nazariyasi (queueing theory) resurslarni dinamik boshqarish tizimlarini modellashtirish uchun keng qo‘llaniladigan matematik asosdir [5]. Kendall notatsiyasi (A/B/C/K/N/D) navbat modellarini standart tarzda tavsiflovchi usul bo‘lib, M/M/1 eng oddiy Puasson-asosli model hisoblanadi. Bulutli muhitdagi elastik ilovalar stsenariysi G/G/n navbat modeli orqali formallashtirilishi mumkin, bunda n o‘zgaruvchan serverlar soni. Little qonuniga ko‘ra tizimda o‘rtacha mijozlar soni E[C] = lambda * E[T] tenglamaga asoslanadi. Ushbu qonun navbatdagi o‘rtacha kutish vaqtini va zarur server sonini aniqlashda keng qo‘llaniladi [5, 3].

Resurslarni taqsimlash (Resource Allocation) – mavjud hisoblash resurslarini foydalanuvchi ilovalari va bajarilayotgan vazifalar o‘rtasida optimal tarzda taqsimlash jarayonidir. Ko‘plab tadqiqotlarda ushbu masala ko‘p mezonli optimallash muammosi sifatida qaraladi. Bunda bajarilish vaqti, energiya sarfi, iqtisodiy xarajatlar hamda SLA buzilishi ehtimoli birgalikda hisobga olinadi [24, 25]. Gorizontal masshtablash ilova instansiyalarining sonini oshirish yoki kamaytirish orqali amalga oshirilsa, vertikal masshtablash mavjud instansiyaga qo‘shimcha CPU, xotira yoki saqlash resurslarini ajratishni nazarda tutadi [3, 21].

4.2 Avto-masshtablash algoritmlari

Avto-masshtablash mexanizmlari bulutli va oqimli hisoblash muhitlarida resurslarni dinamik boshqarishning asosiy tarkibiy qismi hisoblanadi [21]. Ushbu mexanizmlar tizim yuklamasidagi o‘zgarishlarga mos ravishda hisoblash resurslarini avtomatik ravishda oshirish yoki kamaytirishni ta'minlaydi. Avto-masshtablashning asosiy maqsadi SLA talablarini buzmasdan xizmat sifatini saqlab qolish hamda resurslardan foydalanish samaradorligini oshirishdan iborat. Avto-masshtablash jarayoni MAPE (Monitor-Analyze-Plan-Execute) tsikliga mos keladi va to‘rt asosiy bosqichdan – monitoring, tahlil, rejalashtirish va ijro etish – iborat [3].

Reaktiv avto-masshtablash algoritmlari joriy tizim holatini monitoring qilish asosida qaror qabul qiladi [21]. CPU yuklamasi, xotira bandligi yoki tarmoq trafiki kabi ko‘rsatkichlar oldindan belgilangan chegaraviy qiymatlardan oshganda yangi instansiyalar ishga tushiriladi, yuklama pasayganda instansiyalar soni kamaytiriladi. Masalan: 'Agar CPU yuklamasi 70% dan yuqori bo‘lsa, 5 daqiqa davomida, ikkita instansiyani qo‘sh va 10 daqiqa harakatlanma' kabi qoidalar asosida ishlaydi [3].

Amazon EC2, RightScale va boshqa bulut platformalarida aynan ushbu turdagi chegaraviy qoidalar standart sifatida taqdim etiladi. Reaktiv yondashuvlarning asosiy afzalligi ularning soddaligi va amaliyotda oson joriy etilishidadir. Biroq reaktiv yondashuvning asosiy kamchiligi – resurslar faqat yuklama oshgandan so‘ng ajratiladi. Natijada qisqa muddatli yuklama sakrashlari paytida tizim kechikishlari ortishi va SLA buzilishi ehtimoli yuzaga keladi [21]. Chegaraviy qiymatlarning noto‘g‘ri sozlanishi tizim tebranishiga (oscillation) – ya'ni instansiyalar sonining ketma-ket oshib-kamayib turishiga – olib kelishi mumkin [3].

Proaktiv avto-masshtablash algoritmlari kelajakdagi yuklamani oldindan bashorat qilishga asoslanadi [21]. Bu yondashuv tarixiy monitoring ma’lumotlari – CPU yuklamasi, xotira bandligi, tarmoq trafigi –

tahlil qilinib, resurs ehtiyoji taxmin qilinadi va zarur resurslar yuklama oshishidan oldin ajratiladi. Proaktiv yondashuv VM ishga tushirish vaqtini hisobga olishi jihatidan ham muhim: yangi virtual mashina mavjud bo‘lishi uchun bir necha daqiqa talab etilishi mumkin [3].

Bulut resurslarini ta'minlash uchun vaqt qatorlari bashorat algoritmlarining chiqishini yaxshilash [22] ushbu yo‘nalishdagi muhim tadqiqot hisoblanadi. Ushbu usul cloud resurslarini ta'minlash aniqligini sezilarli darajada oshiradi va SLA buzilish ehtimolini kamaytiradi. Ko‘p o‘lchamli vaqt qatorlari asosidagi yondashuvlar [27] bir vaqtning o‘zida bir nechta metrikani – CPU, xotira, tarmoq trafigi – hisobga olib, bashorat aniqligini oshiradi. Biroq bashorat xatolari noto‘g‘ri masshtablash qarorlariga olib kelishi mumkin, bu esa proaktiv yondashuvlarning asosiy cheklovlaridan biri hisoblanadi.

Gibrid avto-masshtablash algoritmlari reaktiv va proaktiv yondashuvlarning afzalliklarini birlashtiradi [21]. Proaktiv modul uzoq muddatli bashoratni amalga oshirsa, reaktiv modul qisqa muddatli keskin o‘zgarishlarga tezkor javob beradi. Tadqiqotlar shuni ko‘rsatadiki, gibrid yondashuvlar mikroservis arxitekturalarida yuqori barqarorlikni ta'minlab, resurslardan foydalanish samaradorligini oshiradi va SLA buzilishini kamayishi mumkin [21]. Ali-Eldin va boshqalar tomonidan taklif etilgan adaptiv gibrid elastiklik kontrolleri [3] ushbu yondashuv bo‘yicha yaxshi misol hisoblanadi: scale up uchun reaktiv, scale down uchun esa proaktiv nazorat qo‘llaniladi.

Mikroservis arxitekturalari uchun mo‘ljallangan resurs-samarali reaktiv va proaktiv avto-masshtablash tizimi [21] ushbu yo‘nalishda muhim qadam hisoblanadi. Container asosidagi bulutli hisoblash muhitida ko‘p mezonli vazifalarni rejalashtirish algoritmi (VBDPA) [24] ham gibrid yondashuvning amaliy namunasi sifatida bajarilish vaqti, energiya sarfi va SLA buzilishini birgalikda minimallashtiradi. Private cloud uchun resurs taqsimlash va foydalanish to‘g‘risidagi real ma’lumotlar to‘plami [26] gibrid avto-masshtablash algoritmlarini sinovdan o‘tkazish uchun qimmatli manba hisoblanadi.

4.3 Resurslarni boshqarishda iqtisodiy jihatlar

Bulutli hisoblashning asosiy iqtisodiy afzalliklaridan biri – to‘g‘ridan-to‘g‘ri operatsion xarajatlarga o‘tish (CapEx dan OpEx ga) imkonidir [2]. Foydalanuvchilar zarur bo‘lganda resurslarni kiralaydi va foydalanmayotganda ozod qiladi. Bu elastiklik kichik kompaniyalarga katta kapital investitsiyalarsiz o‘z bizneslarini tez kengaytirish imkonini beradi. Animoto kabi real dunyo misollari ham ushbu elastiklikning muhimligini ko‘rsatadi: platforma uch kunda 50 serverdan 3500 serverga ko‘tarilishga majbur bo‘lgan va aynan bulutli elastiklik bu surur oshishni moliyaviy jihatdan mumkin qilgan [2].

Resurslarni haddan tashqari ko‘p ajratish (over-provisioning) va kam ajratish (under-provisioning) ikkala holat ham iqtisodiy ziyon keltiradi. Over-provisioning holda foydalanilmayotgan resurslar uchun to‘lov amalga oshiriladi, under-provisioning holda esa xizmat sifatining yomonlashuvi foydalanuvchilarni yo‘qotishga olib keladi [2]. Shu sababli avto-masshtablash algoritmlari faqat texnik emas, balki iqtisodiy optimal yechim izlashi lozim. Xarajatlarni minimallashtirishga qaratilgan avto-masshtablash tizimi [21] aynan shu muvozanatni topishga harakat qiladi – xizmat sifatini buzmasdan xarajatlarni kamaytirish.

Bulutli resurslarni ta'minlash xarajatlari ko‘p omilga bog‘liq: VM turi va hajmi, foydalanish soatlari soni, ma’lumot uzatish hajmi va qo‘shimcha xizmatlar. Container asosidagi muhitlarda resurs granularlikni boshqarish yanada osonlashgan – Docker konteynerlar VM larga qaraganda tezroq ishga tushadi va resurslarni mikrodarajada ajratish imkonini beradi [6]. Bu holat cloud-native ilovalar uchun avto-masshtablash samaradorligini yanada oshiradi.

1-jadval. Avto-masshtablash algoritmlarining qiyosiy tahlili

Mezon	Reaktiv	Proaktiv	Gibrid
Qaror qabul qilish asosi	Joriy monitoring	Yuklama bashorati	Monitoring + bashorat
Javob berish tezligi	O‘rtacha	Yuqori	Yuqori
SLA buzilish ehtimoli	Yuqori	Past	Juda past
Bashoratga bog‘liqlik	Yo‘q	Yuqori	O‘rtacha
Hisoblash murakkabligi	Past	O‘rtacha	Yuqori
Amaliy joriy etish	Oson	O‘rtacha	Murakkab
Kechikishga sezgir ilovalar	Past moslashuv	O‘rtacha moslashuv	Yuqori moslashuv

5 MASHINA O‘RGANISHGA ASOSLANGAN YONDASHUVLAR

So‘nggi yillarda mashina o‘rganishi (ML) texnologiyalarining jadal rivojlanishi resurslarni dinamik taqsimlash va avto-masshtablash jarayonlarini avtomatlashtirishda yangi imkoniyatlarni ochmoqda [25, 32]. ML asosidagi yondashuvlar an'anaviy qoidaga asoslangan mexanizmlarga nisbatan murakkab bog‘lanishlarni aniqlash, noaniqlik sharoitida qaror qabul qilish hamda moslashuvchan boshqaruvni amalga oshirish qobiliyatiga ega [22, 27]. ML yondashuvlari odatda uch asosiy guruhga bo‘linadi: vaqt qatorlari usullari, chuqur o‘rganish modellari va mustahkamlovchi o‘rganish.

5.1 Vaqt qatorlari asosidagi yondashuvlar

Vaqt qatorlari asosidagi yondashuvlarda tizimning tarixiy monitoring ma'lumotlari – CPU yuklamasi, xotira bandligi va tarmoq trafigi – tahlil qilinib, kelajakdagi resurs ehtiyoji bashorat qilinadi. ARIMA (Auto-Regressive Integrated Moving Average), SARIMA va regressiya modellariga asoslangan yondashuvlar bulutli resurslarni ta'minlash uchun keng qo‘llanilmoqda [22]. Ushbu modellar trend va mavsumiylikni aniqlashda samarali bo‘lsa-da, murakkab nolinchi bog‘lanishlarni to‘liq ifodalay olmasligi mumkin.

Bulut resurslarini ta'minlash uchun vaqt qatorlari bashorat algoritmlarining chiqishini yaxshilash [22] ARIMA-ga nisbatan aniqlikni sezilarli oshiradigan yangi usullarni taklif etadi. Ko‘p o‘lchamli vaqt qatorlari (multi-time series) asosidagi yondashuvlar [27] bir vaqtning o‘zida bir nechta metrikani hisobga olib, bashorat aniqligini oshiradi. Natijada proaktiv avto-masshtablash mexanizmlarining samaradorligi sezilarli darajada yaxshilanadi hamda SLA buzilish ehtimoli kamayadi. Cloud resurslaridan foydalanishni yaxshilash uchun optimallashtirilgan ko‘p vaqtli qatorlar bashorati [27] annual bashorat xatosini kamaytirishda yangi natijalarga erishadi.

ML asosidagi ish yukini bashorat qilish va oqimli ilovalar ofloadingini samarali amalga oshirish uchun vazifalarni klasterlash tizimi – ML WPStreamCloud [18] – ish yukini oldindan bashorat qilib, vazifalarni klasterlash orqali cloud resurslarini ta'minlash samaradorligini oshirishda muhim rol o‘ynaydi. Ushbu tizim stream ilovalarini ofloading qilishning xarajatlarini kamaytirish va SLA talablarini bajarish uchun mo‘ljallangan.

5.2 Chuqur o‘rganish modellari

Chuqur o‘rganish modellari (LSTM, GRU, CNN) murakkab va nolinchi bog‘lanishlarni o‘rganish qobiliyati bilan ajralib turadi. LSTM (Long Short-Term Memory) va GRU (Gated Recurrent Unit) modellari vaqt qatorlaridagi uzoq muddatli bog‘liqliklarni aniqlashda ayniqsa samarali. LSTM asosidagi bashorat modellaridan foydalanish proaktiv avto-masshtablash aniqligini sezilarli oshirishi ko‘plab ilmiy ishlarda isbotlangan [22, 27].

Cloud ma'lumot markazlarida energiya samarali vazifalarni rejalashtirish uchun DRL ko‘plab tadqiqotlarda qo‘llanilmoqda [33]. Future Generation Computer Systems jurnalida chop etilgan tadqiqotda DRL asosidagi vazifalar rejalashtirgichi enerija sarfini kamaytirish, vazifa bajarish foizini oshirish va boshqa baseline modellarga nisbatan ustunligini ko‘rsatgan [33]. Biroq chuqur o‘rganish modellarini o‘qitish uchun katta hajmdagi sifatli ma'lumotlar talab etiladi va model xarajati nisbatan yuqori. Shu sababli kam ma'lumotli sharoitlarda chuqur o‘rganish modellarini qo‘llash cheklangan bo‘lishi mumkin.

5.3 Mustahkamlovchi o‘rganish yondashuvlari

Mustahkamlovchi o‘rganish (RL) yondashuvlarida agent muhit bilan o‘zaro ta'sir orqali optimal boshqaruv siyosatini shakllantiradi [25]. Agent har bir qabul qilingan qaror natijasida mukofot (reward) oladi va vaqt o‘tishi bilan eng samarali strategiyani o‘rganadi. Q-learning – RL algoritmlarining eng keng tarqalgan turi bo‘lib, Q-qiymat funksiyasi (action-value function) orqali har bir holat uchun eng yaxshi amalni aniqlaydi.

Chuqur mustahkamlovchi o‘rganish (DRL) algoritmlari resurslarni taqsimlash, vazifani boshqa muhitga uzatish va marshrutlash muammolarini hal qilishda yuqori samaradorlik ko‘rsatmoqda [25, 29]. SDN tarmoqlarida DRL asosidagi marshrutlash algoritmi yo‘qotishlar, kechikish va kengligi kafolatlangan holda yuqori unumdorlikka erishilganini ko‘rsatadi [29]. Cloud hisoblashda ish rejalashtirilishi va resurslarni boshqarish uchun DRL metodlarining keng ko‘lamli sharhi [32] turli DRL algoritmlarining metodologiyasi, ishlash ko‘rsatkichlari va amaliy qo‘llanishini tahlil qiladi.

Resurs taqsimlash va marshrutlash muammolari uchun ko‘p maqsadli DRL algoritmida resurs taqsimotini klasterlash usuli qo‘llanilib, ish vaqti va xarajatlarning sezilarli kamayishi isbotlangan [25]. Ushbu yondashuv muhit dinamik bo‘lgan sharoitlarda ham moslashuvchan boshqaruvni ta'minlay oladi. Biroq DRL algoritmlarida o‘qitish jarayonining murakkabligi, katta holat fazosi va o‘rganish vaqtining uzoqligi asosiy cheklovlar hisoblanadi. Ko‘p kirish Edge Computing (MEC) muhitida resurs rejalashtirilishi uchun RL va DRL texnikalariga e'tiborni qaratgan tadqiqot [Cluster Computing, 2025] 2019-2023-yillar oralig‘idan 97 ta RL-asosli yechimni o‘rganib, eng muhim tendentsiyalarni aniqlagan.

5.4 Energiya-samarali resurslarni boshqarish

Energiya samaradorligi zamonaviy bulutli resurslarni boshqarish tizimlarida tobora muhim jihat bo‘lib bormoqda. Ma'lumotlar markazlari global elektr energiyasi iste'molining taxminan 1-2% ni tashkil etadi va ushbu ko‘rsatkich AI va big data hisoblash talablarining oshishi bilan yildan-yilga o‘smoqda. Yashil

hisoblash (green computing) tamoyillariga asoslangan resurs boshqaruvi algoritmlari nafaqat xarajatlarni, balki ekologik izni ham kamaytiradi [24, 33].

DRL asosidagi energiya-samarali vazifalar rejalashtirish tizimi [33] bulutli muhitda energiya sarfini minimallashtirishga qaratilgan. Ushbu tizim vazifalar bajarilish vaqtini, energiya sarfini va QoS ko‘rsatkichlarini birgalikda hisobga oladigan mukofot funksiyasini o‘rganadi. Natijalar DRL asosidagi tizimning an'anaviy algoritmlarga nisbatan sezilarli energiya tejamkorligini ta’minlaganini ko‘rsatadi. Bir vaqtda band bo‘lmagan server quvvatining ikki uchdan bir qismini iste'mol qilishi [2] energiya tejash bo‘yicha yanada amaliy choralar ko‘rishni talab etadi.

Aqlli energiya boshqaruvida Edge-Fog-Cloud gibrid tizimi va metaheuristik optimallashtirish asosidagi bashorat [20] tizimning energiya samaradorligini oshirishda yangi imkoniyatlar yaratmoqda. Ushbu tizim Edge va Fog qatlamlarida energiya sarfini real vaqtda kuzatib boradi va bashorat asosida resurslarni oldindan sozlaydi. Bunday yondashuv ayniqsa aqlli binolar, quyosh energiyasi tizimlari va sanoat avtomatikasida keng qo‘llanilmoqda. Resurslarni boshqarish tizimlarini energiya samaradorligi mezonlari bilan integratsiyalash kelgusidagi tadqiqotlarning ustuvor yo‘nalishlaridan biri hisoblanadi.

6 VAZIFALARNI REJALASHTIRISH VA RESURSLARNI TAQSIMLASH

Bulutli va klaster hisoblash muhitlarida resurslarni dinamik taqsimlash jarayoni vazifalarni qaysi hisoblash tugunida va qanday ketma-ketlikda bajarilishini belgilovchi vazifalarni rejalashtirish algoritmlari bilan uzviy bog‘liqdir [23, 10]. Ushbu ikki jarayonning integratsiyalashgan holda tashkil etilishi tizimning umumiy ishlash unumdorligini, kechikish darajasini hamda resurslardan foydalanish samaradorligini sezilarli darajada oshiradi.

6.1 Klassik va evristik algoritmlar

Klassik rejalashtirish algoritmlari – FCFS (First Come First Serve) va Round Robin (RR) – operatsion tizim darajasida qo‘llaniladi. Evristik taqsimlash algoritmlari – Min-Min va Max-Min – asosan taqsimlangan va bulutli muhitlarda qo‘llaniladi: Min-Min eng kichik bajarilish vaqtiga ega vazifani birinchi joylashtirsa, Max-Min eng katta bajarilish vaqtiga ega vazifani ustuvor joylashtiradi. Mazkur evristik yondashuvlar vazifalarni resurslar o‘rtasida samarali taqsimlashga xizmat qilsa-da, ular statik xarakterga ega bo‘lib, tizim holatining dinamik o‘zgarishlarini to‘liq hisobga ololmaydi [23].

MapReduce modelida vazifalarni rejalashtirish Hadoop taqsimlangan fayl tizimi (HDFS) bilan birgalikda ishlaydi. MapReduce asosiy kontributorlaridan biri – tarmoq kengligi resursini tejash uchun ma’lumotlar yaqinida hisoblash (data locality) prinsipi bo‘lib, bu sezilarli tarmoq tejamkorligini ta’minlaydi [1]. Sekinlovchi vazifalarni (stragglers) aniqlash uchun klassifikatsiyaga asoslangan model [28] Hadoop klasterida tizim o‘tkazuvchanligini oshirishga yordam beradi: turli xil bo‘lgan Hadoop klasterida straggler vazifalarni bartaraf qilish va boshqarish freymvorki kechikish muammosini hal etishda samarali.

6.2 Dinamik rejalashtirish

Dinamik muhitlarda resurs taqsimlash va rejalashtirish masalalari real vaqt rejimida qaror qabul qilishni talab etadi. Bunday yondashuvlarda tizimning joriy holati – tugunlar yuklamasi, mavjud resurslar va vazifa xususiyatlari – uzluksiz monitoring qilinadi va rejalashtirish qarorlari mos ravishda yangilanadi [23]. Autonomous va dinamik resurs taqsimlash bilan integratsiyalashgan vazifalarni rejalashtirish algoritmi [23] umumiy bajarilish vaqtini va kechikishni sezilarli darajada kamaytiradi.

Petri Net asosidagi freymvork [30] Edge Cloud Continuum muhitida resurslarni rejalashtirish siyosatlarini modellashtirish va simulyatsiya qilishda qo‘llaniladi. Bu yondashuv turli rejalashtirish strategiyalarini solishtirish va optimalini tanlashga imkon beradi. Taqsimlangan va bulutli muhitlarda resurslarni samarali boshqarish bo‘yicha tadqiqotlar [23] ushbu sohadagi ilmiy bilimlarni sezilarli kengaytirgan.

Cloud hisoblashda ish rejalashtirilishi va resurslarni boshqarish uchun DRL metodlarining ko‘lamli sharhi [32] ushbu sohadagi turli algoritmlarni taqqoslab, eng ko‘p va'da beruvchi yo‘nalishlarni belgilaydi. Xususan, DRL asosidagi tizimlar an'anaviy evristik algoritmlardan oshib ketadigan performance ko‘rsatkichlarini namoyish etgan [32].

6.3 Ko‘p mezonli optimallash

Ko‘p mezonli optimallash (multi-objective optimization) yondashuvi bajarilish vaqti, energiya sarfi, kechikish va iqtisodiy xarajatlar kabi mezonlarni birgalikda hisobga oladi [24, 25]. Resurs taqsimotini klasterlash orqali ko‘p maqsadli vazifalarni ofloading optimallash uchun DRL [25] ushbu yo‘nalishda

muhim hissa qo‘shgan. Ushbu yondashuv turli resurs cheklovlari va kechikish talablari mavjud murakkab muhitlarda ham moslashuvchan qarorlar qabul qilish imkonini beradi.

Container asosidagi bulutli hisoblash muhitida ko‘p mezonli vazifalarni rejalashtirish algoritmi (VBDPA) [24] bajarilish vaqti, energiya sarfi va SLA buzilishini bir vaqtda minimallashtirish maqsadiga qaratilgan. Ushbu algoritm Pareto fronti asosida muvozanatli yechimlar to‘plamini aniqlaydi. Private cloud uchun resurs taqsimlash va foydalanish to‘g‘risidagi ma’lumotlar to‘plami [26] real eksperimental tadqiqotlar uchun muhim manba bo‘lib, algoritmlarni real sharoitda sinovdan o‘tkazishga imkon beradi.

6.4 Rejalashtirish va kechikish o‘rtasidagi bog‘liqlik

Vazifalarni rejalashtirish va kechikishni optimallashtirish masalalari bir-biridan ajralmas ekanligi ko‘plab tadqiqotlarda isbotlangan. Yaxshi tuzilgan rejalashtirish mexanizmlari nafaqat resurslardan foydalanish samaradorligini oshiradi, balki umumiy tizim kechikishini ham kamaytiradi. Jumladan, MapReduce arxitekturasida data locality prinsipi – hisoblashni ma’lumotlar saqlangan mashinaga yaqinroq joylashtirish – tarmoq trafigini kamaytirish va shu orqali kechikishni qisqartirish imkonini beradi [1].

Kechikishni hisobga oluvchi operator joylashishi [9] stream processing tizimlari uchun end-to-end kechikishni minimallashtiradigan operator joylashtirish strategiyasini topadi va standart rejalashtirish strategiyalari bilan solishtirganda javob vaqtini sezilarli kamaytiradi. Edge Cloud Continuum muhitida Petri Net asosidagi model [30] turli rejalashtirish siyosatlarini turli ish yuki stsenariylari ostida baholash uchun qulay vosita hisoblanadi. Strategik vazifalarni uzatish masalasi ham rejalashtirish bilan uzviy bog‘liq: Edge tugunlariga qanday vazifalarni, qachon va qancha vaqt uchun uzatish kerakligi to‘g‘ri belgilanmasa, tarmoq yuklanishi ortishi va umumiy kechikish oshishi mumkin [19].

MapReduce klasterlarida ma’lumotlarning qiyshiqlik muammosi (content popularity skew) ham rejalashtirish samaradorligiga bevosita ta'sir ko‘rsatadi. Microsoft Bing Cosmos klasterida mahsulot ishlarini kuzatish natijalariga ko‘ra, mashhur ma’lumotlar replikatsiyasiz real vaqt ilovalarida jiddiy samarasizlikka olib kelishi mumkin [4]. Bunday holatlarda Scarlett kabi proaktiv replikatsiya tizimlari mavjud resurslardan oqilona foydalanib, ish bajarilish vaqtini real ishlab chiqarish klasterlarda sezilarli kamaytiradi. Hadoop muhitida Scarlett map vazifalarini mahalliy muhitda bajarish ulushini 57% dan 83% gacha oshirib, ish bajarilish vaqtini 20,2% ga tezlashtirgani isbotlangan [4].

7 QIYOSIY TAHLILI

Katta ma’lumotlar oqimini qayta ishlash va resurslarni boshqarish sohasida taklif etilgan yondashuvlar turli darajalarda hamda turli texnologik asoslarda ishlab chiqilgan. Ayrim yondashuvlar tizimning joriy holatiga moslashishga yo‘naltirilgan bo‘lsa, boshqalari kelajakdagi yuklamani bashorat qilishga yoki qaror qabul qilish jarayonini avtomatlashtirishga qaratilgan. Qiyosiy tahlil ularning qaysi sharoitlarda samaraliroq ekanligini aniqlash imkonini beradi (2-jadval).

2-jadval. Oqimli qayta ishlash tizimlari va resurslarni boshqarish yondashuvlarining qiyosiy tahlili

Yondashuv turi	Asosiy g‘oya	Afzalliklari	Cheklovlari
Lambda arxitekturasi	Batch + real vaqt qatlamlar	Yuqori ishonchlilik, fault tolerance	Yuqori tizim murakkabligi
Kappa arxitekturasi	Yagona stream qatlam	Soddalik, past kechikish, bitta kod bazasi	Pahalı qayta ishlash
Edge-Cloud ofloading	Ma’lumot manbai yaqinida ishlash	Past javob vaqti, tarmoq tejash	Boshqaruv murakkab
Reaktiv avto-masshtablash	Joriy yuklamaga asoslanish	Soddalik, tezkorlik	Kech javob, SLA buzilishi
Proaktiv avto-masshtablash	Yuklamani bashorat qilish	Oldindan tayyorlanish, SLA kafolati	Bashorat xatosiga sezgir
Gibrid avto-masshtablash	Reaktiv + proaktiv	Yuqori barqarorlik, kam SLA buzilishi	Amalga oshirish murakkab
Vaqt qatorlari ML	Tarixiy ma’lumotlar bashorat	Nisbatan oson amalga oshirish	Murakkab boglanishni qiyin
Chuqur o‘rganish DRL	Mukofotga asoslangan o‘rganish	Avtomatik, moslashuvchan, yuqori aniqlik	O‘qitish murakkab va uzoq
Ko‘p mezonli rejalashtirish	Bir nechta mezoni hisobga olish	Balansli optimallash, real yechim	Lokal optimumga tushib qolish

Amaliy tatbiq etish nuqtai nazaridan qaraganda, Lambda arxitekturasini qo‘llab-quvvatlagan PROMENADE platformasi [6] va WALLeSMART [8] kabi tizimlar ushbu arxitekturani real sharoitda

samarali ekanligi isbotlandi. PROMENADE Lyon shahridagi 317,000 dan ortiq yo‘l segmentini real vaqtda monitoring qilib, o‘rtacha 25 daqiqa kechikish bilan betweenness centrality ko‘rsatkichlarini hisoblab beradi. WALLeSMART esa Belgium Walloniya mintaqasidagi 30 ta fermer xo‘jaligi va 45 ta agrometerologik stansiyadan ma’lumotlarni qayta ishlab, real vaqt ishlari uchun o‘rtacha 1 daqiqa 18 soniyalik bajarilish vaqtiga erishadi [8]. Bu natijalar bulutli arxitektura va zamonaviy stream processing vositalarini birgalikda qo‘llashning amaliy samaradorligini yaqqol ko‘rsatadi.

Resurslarni boshqarish algoritmlari sohasida ham o‘ziga xos tendentsiyalar kuzatilmoqda. Threshold-based (chegaraviy qoidalarga asoslangan) reaktiv yondashuvlar Amazon EC2, RightScale kabi platformalarda standart sifatida amalga oshirilgan bo‘lsa, yangi avlod platformalar tobora ko‘proq ML asosidagi proaktiv va gibrid yondashuvlarga o‘tmoqda. Bulutli resurslarni ta’minlash uchun vaqt qatorlari bashoratini yaxshilash [22] va ko‘p o‘lchamli vaqt qatorlari bashoratini optimallash [27] ushbu o‘tish jarayonida muhim rol o‘ynaydi. DRL asosidagi yondashuvlar esa bir qancha muhim sohalar – SDN marshrutlash [29], energiya-samarali vazifalar rejalashtirish [33], ko‘p maqsadli ofloading optimallash [25] – bo‘yicha an'anaviy algoritmlardan oshib ketadigan natijalarni namoyish etmoqda.

Ushbu tahlil asosida shuni ta'kidlash lozimki, oqimli qayta ishlash tizimlarida kechikishni kamaytirish va resurslarni dinamik boshqarish masalalarini alohida ko‘rish to‘g‘ri emas. Kechikishni kamaytirish ko‘pincha resurs sarfini oshirishni talab qiladi, resurslarni tejash esa kechikishning ortishiga olib kelishi mumkin. Shu sababli optimal yechim doimo ushbu ikki maqsad o‘rtasida muvozanatni topishdan iborat. Zamonaviy ko‘p mezonli optimallash algoritmlari [24, 25] aynan mana shu muvozanatni avtomatik va dinamik tarzda topish uchun mo‘ljallangan.

Arxitektura darajasida Lambda arxitekturasi ishonchlilik va xatoliklarga bardoshlilikni ta’minlasa-da, yuqori tizim murakkabligini keltirib chiqaradi va ikki alohida qayta ishlash tizimini saqlashni talab etadi [3, 8]. Kappa arxitekturasi dizaynni soddalashtiradi va bitta kod bazasini ta’minlaydi, lekin tarixiy ma’lumotlarni qayta ishlash paytida yuqori hisoblash xarajatlarini keltirib chiqarishi mumkin [4, 12]. Telekommunikatsiya ilovalarida Lambda arxitekturasi hali ham ustun bo‘lib qolmoqda, IoT va real vaqt monitoring ilovalarida esa Kappa arxitekturasi keng ommaboqlashmoqda [13].

Avto-masshtablash bo‘yicha reaktiv yondashuvlar soddaligi bilan ajralib tursa-da, qisqa muddatli yuklama sakrashlari paytida SLA buzilish xavfi mavjud [21]. Proaktiv yondashuvlar aniq bashorat modellariga bog‘liq bo‘lib, bashorat xatosi noto‘g‘ri masshtablashga olib kelishi mumkin. Gibrid yondashuvlar ushbu kamchiliklarning ikkalasini ham qisman bartaraf etib, zamonaviy tadqiqotlarda ustuvor yo‘nalish sifatida ko‘rilmoqda [21].

ML asosidagi yondashuvlar ichida DRL eng moslashuvchan boshqaruv strategiyasini ta’minlash qobiliyatiga ega bo‘lsada, o‘qitish va moslashish uchun ko‘p vaqt va resurs talab etadi [32, 33]. Umuman olganda, eng samarali natijalarga bitta yondashuvning o‘zi emas, balki bir nechta komplementar yondashuvlarni integratsiyalash orqali erishish mumkin. ML asosidagi yuklama bashoratini gibrid avto-masshtablash bilan, ko‘p mezonli optimallashni dinamik rejalashtirish bilan birlashtirish tizim barqarorligini va resurslardan foydalanish samaradorligini eng yuqori darajaga olib chiqadi.

7.1 Amaliy platformalar va vositalar tahlili

Stream processing uchun mavjud ochiq manbali frameworklar – Apache Flink, Apache Storm, Apache Spark Streaming va Kafka Streams – har biri o‘ziga xos afzallik va cheklovlarga ega. Apache Flink event-time ishlov berish va stateful operatsiyalar uchun eng kuchli imkoniyatlarni taqdim etadi va ayniqsa Kappa arxitekturasi bilan mos keladi. Apache Storm esa eng eski va minimal kechikishli framework bo‘lib, WALLeSMART [8] kabi real amaliy tizimlarda qo‘llaniladi. Apache Spark Structured Streaming birlashtirgan batch va stream qayta ishlashni ta’minlab, Lambda arxitekturasi uchun yaxshi asos bo‘ladi.

Ma’lumot broker texnologiyalari orasida Apache Kafka de-facto standart bo‘lib qolgan. Kafka sekundiga millionlab xabarlarni qayta ishlay oladi, ma’lumotlarni uzoq muddatda saqlaydi va bir nechta consumer-larning mustaqil ravishda bir xil ma’lumotlarni o‘qishiga imkon beradi [8]. Bu xususiyat Lambda arxitekturasida batch va tezlik qatlamlari uchun bitta manba bo‘lib xizmat qilishda juda muhim. Kappa arxitekturasida esa Kafka immutable jurnal sifatida barcha ma’lumot hodisalarini tartibli va doimiy ravishda saqlaydi va tarixiy ma’lumotlarni replay qilish imkonini beradi [4, 12].

Bulut xizmat ekotizimlarining taqqoslash ko‘rsatkichlari ham muhim ahamiyat kasb etadi. Amazon AWS, Microsoft Azure va Google Cloud har biri o‘ziga xos stream processing xizmatlarini taqdim etadi: AWS Kinesis, Azure Event Hubs va Google Cloud Pub/Sub mos ravishda. Ma’lumotlar platformalari uchun jarayon-yo‘naltirilgan metodologiya [7] ushbu turli ekotizimlarni tizimli ravishda taqqoslab, ma'lum loyiha talablariga eng mos xizmatlar to‘plamini aniqlashga yordam beradi. Bu yondashuv loyihalovchilarga turli provayder ekotizimlarini ob'ektiv va tarafkashliksiz baholash imkonini beradi, bu esa amaliyotda juda muhim.

7.2 Ochiq muammolar va istiqbolli tadqiqot yo‘nalishlari

Katta ma’lumotlar oqimini qayta ishlash va resurslarni dinamik boshqarish sohalarida sezilarli yutuqlarga erishilgan bo‘lsa-da, hozirgi kunda ham bir qator hal etilmagan muammolar mavjud. Ushbu bo‘limda asosiy ochiq muammolar va istiqbolli tadqiqot yo‘nalishlari belgilanadi.

Birinchidan, mavjud avto-masshtablash algoritmlarining aksariyati ma'lum bir platforma yoki muhit uchun moslashtirilgan bo‘lib, turli hisoblash muhitlari (bulutli, chekka va tumanli) o‘rtasida yagona universal yechim hali shakllanmagan [21]. Ayniqsa, ko‘p muhitli gibrid tizimlarda Edge, Fog va Cloud resurslarni birgalikda boshqaruvchi mexanizmlar ilmiy tadqiqotlarda yetarlicha o‘rganilmagan. Ushbu ko‘p qatlamli muhitlarda resurs taqsimlash NP-hard muammo bo‘lib qolmoqda, bu esa taxminiy va evristik algoritmlarni ishlab chiqishni talab etadi [25].

Ikkinchidan, mashina o‘rganish va chuqur o‘rganishga asoslangan yondashuvlarda modelni o‘qitish uchun katta hajmdagi sifatli ma’lumotlar talab qilinadi. Ma’lumotlar sifati past bo‘lgan hollarda bashorat aniqligi pasayadi [22, 27]. Ish yukidagi o‘zgarishlarga doimiy ravishda moslasha oladigan o‘z-o‘zini o‘rganuvchi (self-adaptive) modellar yaratish dolzarb vazifa bo‘lib qolmoqda. Transfer learning va meta-learning yondashuvlarini cloud resurs boshqaruvi masalasiga qo‘llash istiqbolli yo‘nalish hisoblanadi.

Uchinchidan, oqimli qayta ishlash operatorlarini joylashtirish va vazifalarni ofloading masalalari katta miqyosli va yuqori dinamik muhitlarda hisoblash jihatidan murakkab bo‘lib qolmoqda [9, 19]. Real vaqt qaror qabul qilishga mos tez va optimalga yaqin algoritmlar talab etiladi. Arxitektura darajasida Lambda arxitekturasining murakkabligini va Kappa arxitekturasining pahalı qayta ishlash muammosini birgalikda hal qiluvchi moslashuvchan gibrid arxitekturalar talab etilmoqda [3, 4].

To‘rtinchidan, energiya samaradorligi va ekologik barqarorlik masalalari tobora muhim ahamiyat kasb etmoqda. Ma’lumotlar markazlarining quvvat sarfi global elektr energiyasi iste'molining sezilarli qismini tashkil etadi va bu muammo yildan-yilga keskinlashmoqda. Energiya sarfini minimallashtirishga yo‘naltirilgan yashil hisoblash (green computing) yondashuvlarini resurslarni boshqarish tizimlariga integratsiyalash DRL asosidagi vazifalarni rejalashtirish orqali amalga oshirilmoqda [33]. Ammo bu sohada ham ko‘p ish qilish zarur.

Beshinchidan, xavfsizlik, maxfiylik va ishonchlilik omillari ko‘pincha ikkinchi darajali masala sifatida qaralmoqda [14]. Real tizimlarda ushbu omillarni hisobga olmagan holda samarali va barqaror boshqaruvni amalga oshirish mumkin emas. Edge-Fog-Cloud muhitlarida ma’lumotlar maxfiyligini ta’minlovchi va zamonaviy kiberxavfsizlik talablarini qondiradigan resurs boshqaruvi mexanizmlari ishlab chiqilishi zarur.

Oltinchidan, oqimli qayta ishlash tizimlarida kechikish bashoratini real vaqt optimallash bilan birlashtiradigan yondashuv hali keng ko‘lamda ishlab chiqilmagan. Cloud gaming uchun real vaqt kechikish bashorati [31] bo‘yicha birinchi qadam qo‘yilgan bo‘lsa-da, ushbu yondashuv boshqa sohalarga tatbiq etilishi lozim. Shuningdek, foydalanuvchilarning yuqori darajadagi maqsad-talablarini (intent-based computing) resurs taqsimlash algoritmlariga tarjima qiladigan intellektual tizimlar talab etilmoqda.

Yettinchidan, ko‘p ijtimoiy muhitlarda (multi-tenant) resurslarni adolatli va samarali taqsimlash muammosi ham to‘liq hal etilmagan. Turli ilovalar va foydalanuvchilar o‘rtasida resurslarni adolatli taqsimlash (fairness) va bir vaqtning o‘zida maksimal samaradorlikni ta’minlash – bu ikkala maqsad ko‘pincha bir-biriga zid bo‘lib, muvozanatlash juda qiyin. Quincy va Delay Scheduling kabi yondashuvlar bu muammoga birinchi qadam bo‘lgan [4], lekin universal yechim hali topilmagan.

8 XULOSA

Ushbu tahliliy maqolada katta ma’lumotlar oqimini real vaqtda qayta ishlash tizimlarida kechikishni optimallashtirish va resurslarni dinamik boshqarishga doir zamonaviy ilmiy tadqiqotlar tizimli ravishda ko‘rib chiqildi va tahlil qilindi. Jami 35 ta ilmiy manba – asosida keng qamrovli tahlil amalga oshirildi.

Arxitektura darajasida Lambda, Kappa va gibrid arxitekturalarning xususiyatlari, afzalliklari va cheklovlari batafsil ko‘rib chiqildi. Lambda arxitekturasi ishonchlilik va xatoliklarga bardoshlilik jihatidan afzal bo‘lsa-da, tizim murakkabligini oshiradi. Kappa arxitekturasi yagona stream qayta ishlash mexanizmi orqali tizimni soddalashtiradi. Cloud-native platformalar – PROMENADE, WALLeSMART – va Edge-Fog-Cloud gibrid arxitekturalari esa zamonaviy amaliy ilovalar uchun keng tarqalgan echimlar bo‘lib qolmoqda. Ma’lumotlar platformalari uchun jarayon-yo‘naltirilgan loyihalash metodologiyasi AWS, Azure va Google Cloud kabi turli ekotizimlarda optimal xizmatlar tanlovini avtomatlashtirishga imkon beradi.

Tarmoq va hisoblash darajasidagi kechikish omillarini tahlil qilish shuni ko‘rsatdiki, RDMA texnologiyasi, operator joylashishini optimallashtirish, vazifalarni ofloading va ML asosidagi bashorat yondashuvlari kechikishni kamaytirish uchun eng samarali usullar hisoblanadi. MapReduce klasterlarida ma’lumotlarning qiyshiqlik muammosi (Scarlett tizimlari orqali) va sekinlovchi vazifalar muammosi (backup tasks mexanizmi orqali) samarali hal etilmoqda. Biroq hech bir usul barcha stsenariylar uchun to‘liq yechim bera olmaydi – turli usullarni kompleks tarzda qo‘llash eng maqbul natijani beradi.

Resurslarni dinamik boshqarish sohasida gibrid avto-masshtablash, ML asosidagi bashorat va ko‘p mezonli optimallash algoritmlarini integratsiyalash strategiyasi eng yuqori samaradorlikka erishishda asosiy yo‘nalish sifatida belgilandi. Chuqur mustahkamlovchi o‘rganish (DRL) algoritmlari murakkab va dinamik muhitlarda avtomatik qaror qabul qilish uchun katta imkoniyatlar ochmoqda, lekin o‘qitish murakkabligi va uzoq moslashish davri asosiy cheklov bo‘lib qolmoqda.

Kelgusidagi tadqiqotlar gibrid mashina o‘rganish modellarini ishlab chiqish, real vaqt rejimida tezkor qaror qabul qiluvchi algoritmlar yaratish, ko‘p muhitli boshqaruv mexanizmlarini takomillashtirish hamda xavfsizlik, energiya samaradorligi va ishonchlilikni hisobga oluvchi kompleks resurs boshqaruvi tizimlarini rivojlantirishga qaratilishi lozim. Transfer learning va meta-learning yondashuvlarini cloud resurs boshqaruvi masalasiga qo‘llash, intent-based computing tamoyillarini stream processing tizimlariga tatbiq etish – bularning barchasi istiqbolli yo‘nalishlar sifatida belgilanadi. Ushbu maqolada tahlil qilingan yondashuvlar va ochiq muammolar kelajakdagi tadqiqotlar uchun muhim nazariy va amaliy asos bo‘lib xizmat qilishini umid qilamiz.

Umuman olganda, ushbu tahliliy sharh shuni ko‘rsatadiki, katta ma’lumotlar oqimini real vaqtda qayta ishlash va resurslarni dinamik boshqarish sohalari bir-biridan ajralmas va o‘zaro ta'sir qiluvchi soha bo‘lib, ularning integratsiyalashgan ko‘rinishda ko‘rib chiqilishi zaruratini alohida ta'kidlash lozim. Apache Kafka, Flink, Hadoop va Kubernetes kabi texnologiyalar to‘plami, Lambda va Kappa arxitekturalari, ML asosidagi bashorat va DRL yondashuvlari birgalikda zamonaviy big data tizimlarining asosini tashkil qilmoqda. O‘zbekiston va Markaziy Osiyo mintaqasida ham ushbu texnologiyalarni tadqiq qilish va amaliyotga tatbiq etish zarurati tobora ortmoqda: aqlli shaharlar, raqamli qishloq xo‘jaligi va sanoat avtomatikasi sohalari uchun mahalliy sharoitlarga mos arxitektura va algoritmlarni ishlab chiqish istiqbolli va dolzarb yo‘nalish hisoblanadi.

ADABIYOTLAR

[1] *J. Dean, S. Ghemawat*, "MapReduce: Simplified Data Processing on Large Clusters," Proc. 6th Symp. on Operating System Design and Implementation (OSDI), San Francisco, CA, 2004.
[2] *M. Armbrust et al.*, "A View of Cloud Computing," Communications of the ACM, vol. 53, no. 4, pp. 50-58, 2010.
[3] *T. Lorido-Botran, J. Miguel-Alonso, J.A. Lozano*, "A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments," Journal of Grid Computing, vol. 12, pp. 559-592, 2014.
[4] *G. Ananthanarayanan et al.,* "Scarlett: Coping with Skewed Content Popularity in MapReduce Clusters," Proc. EuroSys 2011, Salzburg, Austria, 2011.
[5] *L. Kleinrock,* Queueing Systems, Volume 1: Theory. New York: John Wiley & Sons, 1975.
[6] *C. Colarusso et al.,* "PROMENADE: A big data platform for handling city complex networks with dynamic graphs," Future Generation Computer Systems, vol. 137, pp. 129-145, 2022.
[7] *M. Francia, M. Golfarelli, M. Pasini,* "Process-driven design of cloud data platforms," Information Systems, vol. 131, p. 102527, 2025.
[8] *A. Roukh et al.,* "Big Data Processing Architecture for Smart Farming," Procedia Computer Science, vol. 177, pp. 78-85, 2020.
[9] *S. Nastic, H. Truong, S. Dustdar*, "Latency-aware placement of stream processing operators in modern-day stream processing frameworks," Future Generation Computer Systems, vol. 95, pp. 208-221, 2019.
[10] *M.M. Rathore, A. Ahmad, A. Paul, G. Jeon*, "A big data platform for smart cities," Future Generation Computer Systems, vol. 101, pp. 101-112, 2019.
[11] *A.I. Maarouf, S. Tata, L. Bellatreche*, "Process-driven design of cloud data platforms," Future Generation Computer Systems, vol. 109, pp. 571-586, 2020.
[12] *A. Zanella et al*., "A new Kappa architecture for IoT data management in smart farming," Computer Networks, vol. 168, 2020.
[13] I. *Afolayan, A.F.S. Dehghantanha, K.-K.R. Choo,* "Big data analytics in telecommunications: Governance, architecture and use cases," J. Network and Computer Applications, vol. 154, 2020.
[14] *Y. Liu et al.,* "Beyond 5G: PHWAN - A secure, low-latency, and cost-effective framework for Industry 4.0 smart manufacturing," Computer Networks, vol. 234, 2023.
[15] *Y. Wang, Z. Liu, H. Zhang*, "Network performance evaluation criterion model for low latency in industrial 50G-PON network," Optical Fiber Technology, vol. 82, 2024.
[16] *S. Kar, P. Mishra, K.-C. Wang*, "Dynamic packet duplication for reliable low latency communication in 5G NR-DC networks," Computer Networks, vol. 234, 2023.
[17] *Y. Zhang, M. Li, K. Zhou, K. You, Y. Yu*, "Design and implementation of an RDMA-based data transmission system prototype," Nuclear Engineering and Technology, vol. 58, 2026.

[18] *M.M. Hassan et al.,* "ML WPStreamCloud: ML-based workload prediction and task clustering for efficient stream application offloading," J. Network and Computer Applications, vol. 201, 2022.
[19] *H. Zhang, Y. Li, X. Chen*, "An adaptive hybrid edge-cloud collaborative offloading method for large-scale computational tasks," Future Generation Computer Systems, vol. 148, pp. 48-61, 2023.
[20] *A. Alqahtani, A. Alabdulatif, A. Alenezi*, "Metaheuristic-optimized forecasting in a smart edge-fog-cloud energy management framework," Energy Reports, vol. 10, pp. 1120-1133, 2024.
[21] *M. Abdelbaky, J. Diaz-Montes, M. Parashar, M. Sadjadi*, "Towards resource-efficient reactive and proactive auto-scaling for microservice architectures," Journal of Systems and Software, vol. 188, 2022.
[22] *J.V.V. Sobral, P.A.D. Domingos, L. Veiga*, "Enhancing the output of time series forecasting algorithms for cloud resource provisioning," Future Generation Computer Systems, vol. 152, pp. 351-365, 2025.
[23] *M.A. Tawfeek et al.,* "An efficient and autonomous dynamic resource allocation in cloud computing with optimized task scheduling," Future Generation Computer Systems, vol. 32, pp. 232-246, 2014.
[24] *R. Singh, P.K. Gupta*, "VBDPA: A multi-criteria task scheduling algorithm in container based cloud computing environment," Journal of Cloud Computing, vol. 11, no. 54, 2022.
[25] *X. Li, Z. Zhao, F. Zhao, H. Zhang*, "Multi-objective task offloading optimization using deep reinforcement learning with resource distribution clustering," Applied Soft Computing, vol. 143, 2023.
[26] *S. Sotiriadis, N. Bessis, E. Asimakopoulou*, "Dataset on resource allocation and usage for a private cloud," Data in Brief, vol. 40, 2022.
[27] *A.A. Alqahtani, A. Alabdulatif, A. Alenezi*, "Optimizing multi-time series forecasting for enhanced cloud resource utilization," Future Generation Computer Systems, vol. 154, pp. 380-394, 2024.
[28] *R. Anwar, M. Rehman, S.K. Das*, "A classification framework for straggler mitigation and management in a heterogeneous Hadoop cluster," J. Parallel and Distributed Computing, vol. 170, pp. 41-55, 2022.
[29] *S. Guo, B. Xiao, Y. Yang,* "DRL-based routing algorithm with guaranteed loss, latency and bandwidth in SDN networks," Computer Networks, vol. 234, 2023.
[30] *J. Li, J. Wu, Z. Zhang*, "A Petri Net-based framework for modeling and simulation of resource scheduling policies in Edge Cloud Continuum," Simulation Modelling Practice and Theory, vol. 132, 2025.
[31] *D. Monaco et al*., "Real-time latency prediction for cloud gaming applications," Computer Networks, vol. 264, 2025.
[32] *G. Zhou, W. Tian, R. Buyya et al*., "Deep reinforcement learning-based methods for resource scheduling in cloud computing: A review and future directions," Artificial Intelligence Review, vol. 57, p. 124, 2024.
[33] *H. Hou, S.N.A. Jawaddi, A. Ismail*, "Energy efficient task scheduling based on deep reinforcement learning in cloud environment," Future Generation Computer Systems, vol. 151, pp. 214-231, 2024.
[34] *M.U. Demirezen, T.S. Navruz*, "Performance Analysis of Lambda Architecture-Based Big-Data Systems on Air/Ground Surveillance Application with ADS-B Data," Sensors, vol. 23, no. 17, p. 7580, 2023.
[35] *A. da Silva Veith et al.,* "Latency-Aware Strategies for Deploying Data Stream Processing Applications on Large Cloud-Edge Infrastructure," IEEE Transactions on Parallel and Distributed Systems, 2021.

Поступила в редакцию 10.10.2025

Citation: *Boynazarov I.M., Mamanov J., Mahmudov J.I., Esonboyev M.* (2026). Katta ma'lumotlar oqimini qayta ishlash: kechikishlarni optimallashtirish va resurslarni dinamik boshqarish algoritmlari tahlili. Raqamli texnologiyalarning nazariy va amaliy masalalari xalqaro jurnali. 9(1). – B. 139-153. https://doi.org/10.62132/ijdt.v9i1.359.

REVIEW OF BIG DATA STREAM PROCESSING: LATENCY OPTIMIZATION AND DYNAMIC RESOURCE MANAGEMENT ALGORITHMS

Boynazarov I.M.[1], *Mamanov J.*[2], *Mahmudov J.I.*[1], *Esonboev M.*[1],

[1] Samarkand Branch of Tashkent University of Information Technologies named after Muhammad al-Khwarizmi, Samarkand, Uzbekistan
[1] Denau Institute of Entrepreneurship and Pedagogy, Denau, Uzbekistan

Abstract. This review paper systematically examines modern architectures, algorithms, and approaches aimed at reducing latency and dynamically managing resources in real-time big data stream processing systems. The characteristics of Lambda, Kappa, and hybrid architectures, cloud-native platforms, as well as Edge–Fog–Cloud hybrid environments are analyzed. Network-level and computation-level latency factors, along with mitigation techniques – including operator placement, task offloading, RDMA technology, and machine learning-based prediction models – are investigated. Dynamic resource management issues in cloud environments are addressed, including reactive, proactive, and hybrid auto-scaling algorithms, time series-based approaches, deep learning and reinforcement learning methods, and multi-objective task scheduling. The analysis demonstrates that the highest efficiency is achievable through the integration of complementary approaches – proactive forecasting, hybrid auto-scaling, and ML-based decision-making. Open research challenges and promising future directions are identified.

Keywords: big data, stream processing, Lambda architecture, Kappa architecture, cloud computing, latency, operator placement, auto-scaling, machine learning, dynamic resource management, Edge–Fog–Cloud.

ОБРАБОТКА ПОТОКОВ БОЛЬШИХ ДАННЫХ: ОПТИМИЗАЦИЯ ЗАДЕРЖЕК И АНАЛИЗ АЛГОРИТМОВ ДИНАМИЧЕСКОГО УПРАВЛЕНИЯ РЕСУРСАМИ

Бойназаров И.М.[1], *Маманов Ж.*[2], *Махмудов Ж.И.*[1], *Эсонбоев М.*[1],

[1] Самаркандский филиал Ташкентского университета информационных технологий имени Мухаммада ал-Хоразмий, Самарканд, Узбекистан
[1] Денауский институт предпринимательства и педагогики, Денау, Узбекистан

Аннотация. В данной обзорной статье систематически рассмотрены современные архитектуры, алгоритмы и подходы, направленные на снижение задержек и динамическое управление ресурсами в системах обработки потоков больших данных в реальном времени. Проанализированы характеристики Lambda-, Kappa- и гибридных архитектур, облачно-нативных платформ, а также гибридных сред Edge–Fog–Cloud. Исследованы факторы задержки на уровне сети и вычислений, а также методы их снижения – включая размещение операторов, выгрузку задач (task offloading), технологию RDMA и модели прогнозирования на основе машинного обучения. Рассмотрены вопросы динамического управления ресурсами в облачной среде: реактивные, проактивные и гибридные алгоритмы автомасштабирования, подходы на основе анализа временных рядов, глубокого обучения и обучения с подкреплением, а также многокритериальное планирование задач. Результаты анализа показывают, что наивысшей эффективности можно достичь путём интеграции различных подходов – проактивного прогнозирования, гибридного автомасштабирования и принятия решений на основе ML. Выявлены нерешённые проблемы и перспективные направления будущих исследований.

Ключевые слова: большие данные, потоковая обработка, Lambda-архитектура, Kappa-архитектура, облачные вычисления, задержка, размещение операторов, автомасштабирование, машинное обучение, динамическое управление ресурсами, Edge–Fog–Cloud.