O‘ZBEKISTON RESPUBLIKASI AXBOROT TEXNOLOGIYALARI VA KOMMUNIKATSIYALARINI RIVOJLANTIRISH VAZIRLIGI

MUHAMMAD AL-XORAZMIY NOMIDAGI TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI URGANCH FILIALI

Ish ko‘rib chiqildi va himoyaga qo‘yildi
Axborot texnologiyalari
kafedrasi mudiri
_____ Xalmuratov O.U.
«____» _____ 2018 yil

Bekmetov G`olibjon Anvar o`g`li

MA`LUMOTLARNI QAYTA ISHLASHDA PARALLEL ALGORITMLARNING SAMARADORLIGINING TAHLILI

mavzusidagi

5330500-“Kompyuter injiniringi” (Kompyuter injiniring) ta’lim yo‘nalishi bo‘yicha bakalavr darajasini olish uchun yozilgan

BITIRUV MALAKAVIY ISHI

Bitiruvchi	_____ (imzo)	Bekmetov G`. (f.i.o.)
Rahbar	_____ (imzo)	Xalmuratov O. (f.i.o.)
Taqrizchi	_____ (imzo)	Raximov I. (f.i.o.)

Urganch – 2018 y.

O ‘ZBEKISTON RESPUBLIKASI AXBOROT TEXNOLOGIYALARI VA KOMMUNIKATSIYALARINI RIVOJLANTIRISH VAZIRLIGI

MUHAMMAD AL-XORAZMIY NOMIDAGI TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI
URGANCH FILIALI

_____ fakulteti _____ kafedrasi

yo‘nalish (mutaxassislik) _____

TASDIQLAYMAN
Kafedra mudiri

_____
2017 «___»_____

Malakaviy bitiruv ishiga

T O P SH I R I Q

Bekmetov G`olibjon Anvar o`g`li

(familiyasi, ismi, otasining ismi)

1. Ish mavzusi: Ma`lumotlarni qayta ishlashda parallel algoritmlarning samaradorligini tahlili.
2. TATU Urganch filialining «20» dekabr 2017 yil sanadagi № 882-XΦ sonli buyruqi bilan tasdiqdangan
3. Ishni himoyaga topshirish muddati: 27.05.2018
4. Ishga oid dastlabki ma’lumotlar: Axborot kommunikatsiya–texnologiyalari izohli lug`ati, “Elektron hukumat to‘g`risidagi” O‘zbekiston Respublikasi Qonuni.
5. Hisoblash-tushuntirish yozuvlarining mazmuni (ishlab chiqiladigan masalalar ro’yxati): Ma`lumotlarni qayta ishlashda parallel algoritmlarning samaradorligini tahlil qilish uchun mavjud kompilyatorlari.
6. Grafik materiallar ro‘yxati: prezentatsiya, grafiklar va sxemelar.
7. Topshiriq berilgan sana: 11.01.2017

Rahbar _____
(imzo)

Topshiriq oldim _____
(imzo)

8. Ishning ayrim bo ‘limlari bo’yicha maslahatchilar

Qism	Maslahatchi o ‘qituvchining F.I.O.	Imzo, sana	
		Topshiriq berildi	Topshiriq olindi
1. Tizimli tahlil va masalaning qo’yilishi	T.O`razmetov	01.02.2018	
2. Asosiy qism	T.O`razmetov	07.03.2018	
Mehnat muhofazasi va texnika xavfsizligi	B. Sabirov	20.04.2018	

9. Ishni bajarish grafigi

№	Ish qismlarining nomi	Bajarish muddati	Rahbar (maslaxatchi) Belgisi
1	Bitiruv ishini tasdiqlash	20.12.2017	
2	Mavzu bo ‘yicha adabiyotlarni yig ‘ish va o ‘rganish	08.01.2018	
3	Tizimli tahlil va masalaning qo‘yilishi (Diplom ishi mavzusi bo’yicha ma’lumotlar yig’ish,saralash va texnik talablar ishlab chiqish)	05.02.2018	
4	Asosiy qism	05.05.2018	
5	Algoritm va programma ta’minoti (Dasturiy ta’minotni ishlab chiqish uchun algoritmni va dastur vositalarni tanlash)	14.05.2018	
6	Mehnat muhofazasi va texnika xavfsizligi (Diplom ishiga mos mehnat muhofazasi va texnika havfsizligi masalalari taxlili)	18.05.2018	
7	Xulosa	21.05.2018	
8	Adabiyotlar ro‘yxati	26.05.2018	
9	Chizma – grafik ishlar, prezentatsiya	28.05.2018	
10	Bitiruv ishini rasmiylashtirish (perepletlash)	30.05.2018	

Bitiruvchi _____ 2018 «_ __»_____ ____
(imzo)

Rahbar _____ 2018 «_ __»_____ ____
(imzo)

KIRISH

Yangi XXI-asrda axborot texnologiyalari hayotimizning turli jabhalariga kirib borishi axborotlashgan jamiyatni shakllantirishga zamin yaratib bermoqda. "Internet", "Elektron pochta", "Elektron ta`lim", "Elektron boshqaruv", "Elektron hukumat", "Masofaviy ta`lim", "Ochiq ta`lim", "Axborotlashgan iqtisod" kabi tushunchalar hayotimizga kirib kelishi jamiyatimizning axborotlashishiga intensiv ta`sir ko`rsatmoqda. Axborot-kommunikatsiya texnologiyalari orqali mamlakatlarning milliy iqtisodi globallashib, axborotlashgan iqtisod shakliga o`tmoqda, ya`ni milliy iqtisoddagi axborot va bilimlarning roli tobora yuksalmoqda va ular strategik resursga aylandi.

XX asrning so`nggi 10 yili mobaynida axborotlar bilan ishlash va axborotlashtirish juda rivojlandi. Bunga sabab shundaki, kundalik turmushda axborotlar, ularni qayta ishlash va uzatishning ahamiyati ortib bormoqda. Bu esa, o`z navbatida jamiyatning xar bir a`zosidan axborotlashtirish va axborot texnologiyalari sirlarini, uning qoida va qonuniyatlarini mukammal bilishni taqozo etadi.

Bitiruv malakaviy ishining dolzarbligi. Bizning davrimizda zamonaviy kompyuterlar qator ustunliklarga ega, lekin zamonaviy kompyuterlarning yuqori hisoblash quvvatiga qaramasdan yechilishi oddiy elektron hisoblash mashina (EHM) larida yo`l qo`yib bo`lmas darajada uzoq vaqt egallaydigan masalalar mavjud.Quyidagi vazifalar bu mavzu dolzarbligining haqqoniy isboti hisoblanadi,bu gidrodinamik jarayonlarni sonli modellashtirish, obrazlarni o`qish vazifalari, ko`p sonli parametrlar bilan muvofiqlashtirish vazifalari va boshqalar.

Tanlangan mavzuning dolzarbligi hisoblash texnikasidan foydalangan holda jarayonlarni matematik modellashtirishda, shuningdek ma`lumotlar bazasi ko`rinishida berilgan turli tabiatli ma`lumotlarni tahlil qilinishida qurilgan ilmiy ishlarning tez rivojlanishining natijasi hisoblanadi.

**Bitiruv malakaviy ishining ob`ekti va predmeti.** Ko`pprotsessorli hisoblash tizimlarida parallel algoritmlarning qo`llanilishi, ma`lumotlarni intellektual tahlillashda tasvirlarni parallel qayta ishlash modeli va algoritmlari.

Bitiruv malakaviy ishining maqsadi. Taqsimlangan muhitda ma`lumotlarni tahlil qilish uchun parallel algoritmlarni qo`llash va samaradorlikni tahlil qilish.

**Bitiruv malakaviy ishida qo`llanilgan metodikaning tavsifi.** Bitiruv malakaviy ishida qo`yilgan vazifalarni hal qilishda Microsoft visual studio kompilyatorining OpenMP va OpenCV muhitini qo`llagan holda, parallel algoritmlardan foydalanib ma`lumotlar tahlilini amalga oshiruvchi dastur yaratish asoslari ko`rib chiqiladi. Bundan tashqari ko`p yadorli protsessorlar hisoblash tizimlarining ishlash tamoyillari qo`llanilgan. Ma`lumotlarni tahlil qilishda protsessorlarning keng imkoniyatlaridan foydalanilgan.

Bitiruv malakaviy ishi natijalarining nazariy va amaliy ahamiyati. Ishlab chiqilgan algoritmik va dasturiy ta`minot quyidagi holatlarda qo`llanilishi mumkin: ma`lumotlarni qayta ishlashda qo`llanilgan parallel algoritmlari va unumdorlik natijalarining tahlili; bir va ko`p yadroli protsessorlarda bajariladigan amallarni oqimlarga ajratish; ko`p yadroli protsessorlarda erishilgan tezkorlikni tahlil qilish va unumdorlikka erishish.

Ish tuzilmasining tavsifi. Bitiruv malakaviy ishi kirish qismidan, uchta bob, xulosa, foydalanilgan adabiyotlar ro`yhati va ilovadan iborat.

I. BOB. MA`LUMOTLARNI INTELLEKTUAL TAHLILLASH, XOTIRA VA PROTSESSOR UNUMDORLIGINI OSHIRISH USULLARI VA PARALLEL HISOBLASHLARNI MODELLASHTIRISH VA ULARNING TAHLILI

1.1. Ma`lumotlarni intellektual tahlillashda tasvirlarni parallel qayta ishlash algoritmlari va texnik vositalari

Pentium mikroprotsessori 64 razryadli magistral asosiga qurilgan bo`lib xotira va registrlardan ma`lumot almashinuvini keskin tezlashtiradi. Bitta bajaruvchi qurilma ikkita U va V qurilmalar bilan almashtirilgan. Ularni xar biri parallel operatsiyalar bajaradi. U qurilmasi asosida xisoblanib barcha buyruqlarni bajaradi. V qurilmasi yordamchi bo`lib faqat eng ko`p uchraydigan buyruqlarni bajaradi. Ichki registr (kesh) ikkiga bo`lingan:

- buyruqlar keshi va
- berilganlar keshi [17].

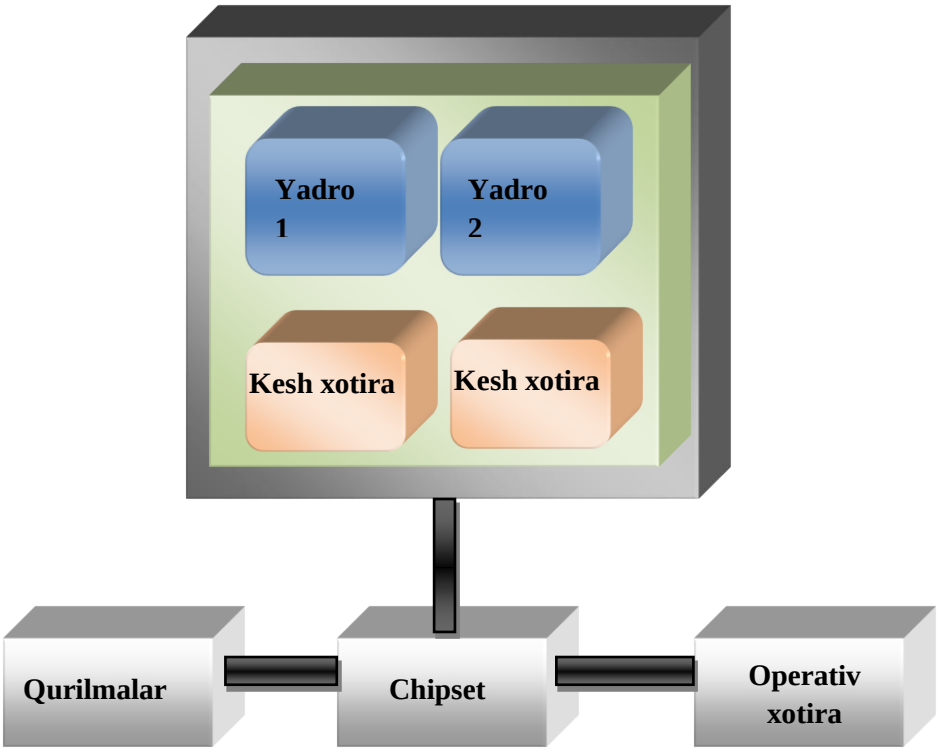
Pentium da ichki chastotada ishlovchi ichki L2 kesh, xajmi 256, 512 yoki 1024 uz kontroller va lokal 64 razryadli berilganlar shina qo`shilgan. Qo`shimcha ichki optimallashtirish, tezlashtirilgan konveyer va parallellashtirish darajasi orttirilgan. Nisbatdan katta quvvatli matematik protsessor o`rnatilgan.

Ma`lumotlarni intellektual tahlil qilishda tasvirlar ustida har xil amallar ketma ketligini bajarishda bir necha bosqichlardan o`tiladi [18,19]. Bular: multimedia axborotini (misol uchun tasvir) xotiradan o`qib olish, bajarilish darajasiga qarab protsessor navbatiga qo`yish, tasvir ma`lumotlarini vaqtinchalik saqlash uchun operativ xotiradan bo`sh joy ajratish va qayta ishlash natijasida olingan tasvir qiymatlarini doimiy xotiraga yozish kabilardan iboratdir.

Ammo bu amallar ketma ketligiga har doim ham aniq vaqt davomida yakunlanmaydi va vaqtdan yutqazishga olib keladi. Bunga bir necha sabablarni misol sifatida ko`rishimiz mumkin.

Birinchidan, eng katta muammolardan biri xotira muammosidir. Tasvir hajmining aynan bir hajmda belgilanmaganligi, tasvirni har safar qayta ishlab, saqlash uchun xotiraga murojat qilinganda yangi hajmda joy ajratishni talab etadi, bu esa tasvirga tegishli qiymatlarni hisoblash, uning kengaytmasini aniqlash va bu o`zgarishlarni operatsion tizim jurnalida qayt etishni va vaqdan yutqazish kabi kamchiliklarga olib keladi. Ikkinchidan, qayta ishlanadigan tasvir protsessorga yuklangach ma`lum chegaralangan vaqt ichida tasvir ma`lumotlari ustida bajariladigan amallar ketma ketligi bajarilib bo`lishi lozim, aks holda katta hajmdagi tasvir ma`lumotlari xotiraga yozish davomida ayrim kechikishlarga olib kelishi mumkin.

Bundan tashqari shinalarni band qilish vaqti ham chegaralangan hisoblanadi. Kompyuter xotirasida tasvirlar hajmining oshib ketishi qayta ishlash tezligining pasayishiga olib keladi. Ma`lumotlarni intellektual tahlil qilish tasvirlarni qayta ishlashda imkon qadar sifat darajasini pasaytirmagan holda kichik hajmda saqlash algoritmlarini ishlab chiqishni talab qiladi.

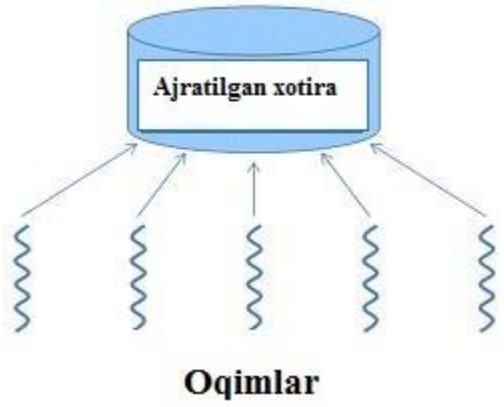


1.1-rasm. Ko`p yadroli ajratilgan kesh-xotirali protsessor

Tasvirlarni qayta ishlashda ko`p yadroli protsessorlar uchun mo`ljallangan parallel qayta ishlash algoritmlardan foydalanish kompyuterning ishlash unumdorligiga ijobiy ta`sirini ko`rsatadi. Ammo oqimlarga ajratish jarayonida tasvir ma`lumotlarining bir qancha qayta ishlashda protsessorga yuklanadigan ma`lumotlar hajmi muhim ahamiyatga ega. Chunki xotiradan vaqtinchalik joy ajratish masalalari har doim dastur unumdorligini oshirishdagi nozik joy hisoblanadi. Xotiradan joy ajratish masalalarini amalga oshiradigan parallelashtirish mexanizmi quyidagi masalalarni yechishi lozim:

- oqim xavfsizligini ta`minlash;
- qo`shimcha xarajatlar;
- musoboqalashtirishni tashkillashtirish;
- xotira nosozliklari.

Ikkita oqim bir vaqtning o`zida xotirani band qilmoqchi yoki uni bo`shatishi kerak bo`lib qolganda, xotirani ajratish mexanizmining ichki ma`lumotlar tuzulishida musoboqalash boshlanadi va dasturning noto`g`ri ishlashiga olib keladi.



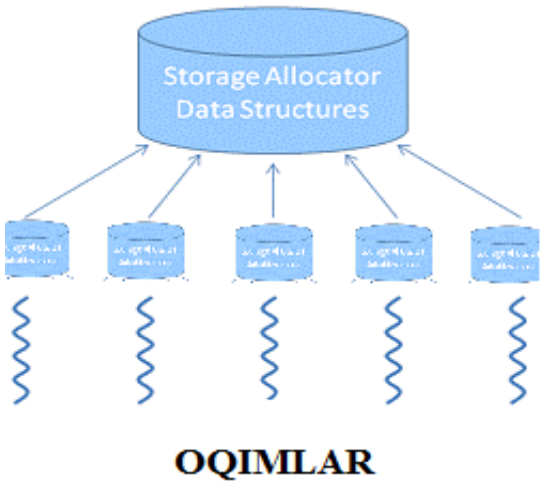
1.2-rasm. Oqimlarning ajratilgan xotiraga murojati

Agar oqimlar 1.2-rasmda ko`rsatilgandek taqsimlangan xotiraga cheklanmagan murojaat qilish huquqiga ega bo`lsalar, unda oqimlar bir vaqtning o`zida aynan bir yacheykaga murojaat qilishlari mumkin, bu esa tizim ishining normadan tashqari ishlashiga olib keladi [21].

Muammoning oddiy yechimi sifatida xotiraga murojaat qilish va undan vaqtinchalik joy ajratishni cheklovchi funksiyalarni tashkil etish, bunda oqimlarning xotiraga murojati bir amal bajarish davomida faqatgina bir marta murojaat qilib olish huquqini beradi. Natijada bir jarayon bajarilish davomida barcha oqimlar xotiraga bir marotaba murojaat qilishadi va barcha oqimlar bajarilishi sinxron tarzda yakunlanadi.

Ajratilgan xotiraga murojaat qilish mexanizmiga taqiqlashlarni joriy etish quyidagi ikkita muammoni yuzaga kelishiga sabab bo`lishi mumkin. Birinchidan, xotiradan joy ajratish va ajratilgan xotirani bo`shatishga taqiqlashlar o`rnatish oqibatida qo`shimcha qutilishlarni sekinlashishiga olib kelishi mumkin.

Ikkinchidan esa, oqimlarning xotiraga ruxsat olishidagi harakati musoboqalashish jarayonlarini yuzaga keltirishi va dastur ishini sekinlashishiga olib kelishi mumkin. Har ikkala muammoni 2.3-rasmda ko`rsatilgandek oqimlarning har biri uchun xotiraga lokal murojaatni tashkil etish bilan yechishimiz mumkin.



1.3-rasm. Har bir oqim uchun ajratilgan lokal xotira

1.3-rasmda keltirilgan texnologiya ajratilgan xotirani taqsimlash deb ataladi. Ajratilgan xotiraga oqimlarni taqsimlab berish ayniqsa tasvir ma`lumotlarni parallel qayta ishlashda yaxshi natija beradi. Parallel qayta ishlashda taqsimlangan tasvir ma`lumotlarning kvadrat matritsalar soniga va ularning o`lchamiga bog`liq hisoblanadi. Matritsalar sonining ko`payishi xotiraga bo`lgan murojatlar sonini

shunchalik oshishiga sabab bo`ladi [22]. Chunki matritsaning o`lchamiga loyiq ajratilgan xotiradan dinamik joy taqsimlash lozim bo`ladi, bu jarayon esa ma`lum miqdorda vaqt sarflanishiga olib keladi. Tasvirlarni raqamli qayta ishlaganda ularni ham protsessor unumdorligini oshirish muammolarini ham xotira bilan bog`liq oqimlarni taqsimlashdagi xatoliklarni bartaraf etishda optimal usullar yaratildi. Ularni bobning davomida amaliyotda qo`llanilgan misollar tariqasi keltirib o`tishimiz mumkin.

Hozirga paytda ma`lumotlarni intellektual tahlil qilishda ya`ni multimedia ma`lumotining bir turi hisoblangan tasvirni siqishning bir necha algoritmlari mavjud, bular ichida JPEG algoritmini misol qilishimiz mumkin. JPEG algoritmining yo`qotishli va yo`qotishsiz siqish usullari mavjud. Yo`qotishli usulda siqishda tasvir hajmi sezilarli kamaymaydi. Tasvir sifat darajasi muhim ahamiyatga ega bo`lmagan hollarda bu usuldan foydalanish yaxshi natija bermaydi va multimedia tizimlarida foydalanganda xotira hajmidan ortiqcha joy egallaydi. Bu muammoni yechishda ko`p yadroli protsessorlar uchun parallellashtirish algoritmlarini qo`llagan holda tasvirlarni yo`qotishli siqishning yana bir yangi algoritmi ishlab chiqildi [23].

Bu algoritmda tasvirni ikki o`lchamli Veyvlet-Dobeshi o`zgartirishi qo`llaniladi. Veyvlet-Dobeshi diskret o`zgartirishda tasvirni siqishgacha bo`lgan amallar ketma-ketligini tezlashtirish maqsadida C++ dasturlash tilining parallellashtirish kutubxonalaridan foydalanildi. Bunda ko`p yadroli protsessorlarda dastur algoritmida bajariladigan arifmetik amallar bir vaqtda har bir yadroga taqsimlab bo`lib berildi. Shundan so`ng Veyvlet – Dobeshi diskret o`zgartirishdan olingan massiv qiymatlari ustida siqish amallarini parallellashtirish algoritmlaridan foydalangan holda amalga oshirildi.

Yuqorida misol sifatida keltirilgan sabablardan protsessor bilan bog`liq tomonlarini oladigan bo`lsak, bunda asosan parallellashtirish amallariga to`xtalsak muammoni yechimiga yaqinlashgan bo`lardik. Parallellashtirishning dasturiy vositalarini oladigan bo`lsak, eng avvalo parallellashtirish texnologiyalari to`g`risida tushunchaga ega bo`lishimiz kerak. Bunga OpenMP, TBB, Intel IPP,

MPI [24] va boshqa dasturiy paketlarni misol qilishimiz mumkin. Lekin bular har biri qo`llanilish soxasiga ko`ra farqlanadi. OpenMP haqida batafsilroq tanishib chiqamiz.

OpenMP texnologiyasining xususiyatlari [25]. OpenMP(Open Multi-Processing) – ko`p oqimli ilovalarni yaratish uchun mo`ljallangan amaliy dasturlashning interfeysi bo`lib, asosan umumiy xotiraga ega bo`lgan parallel hisoblash tizimlari uchun ishlab chiqilgan. OpenMP kompilyatorlar va maxsus funksiyalar bibliotekasi uchun direktivalar to`plamidan iborat. OpenMP standarti yaqin 15 yil ichida umumiy xotiraga ega arxitekturalarga qo`llanilgan holda yaratilgan. So`nggi yillarda taqsimlangan xotirali parallel hisoblash tizimlari uchun OpenMP standartining kengaytirilgan holda ishlab chiqilmoqda. 2005-yilning oxirida Intel kompaniyasi Cluster OpenMP mahsulotini taqdim etdi. Unda kengaytirilgan OpenMP ishlab chiqilgan bo`lib taqsimlangan xotirali parallel hisoblash tizimlari uchun mo`ljallangan.

OpenMP spetsifikatsiyasini hisoblash va dasturlash texnikasi bo`yicha bir nechta yirik ishlab chiqaruvchi kompaniyalar (Intel, Hewlett-Packard, Silicon Graphics, Sun, IBM, Fujitsu, Hitachi, Siemens, Bull) yaratishmoqda, ularni OpenMP Architecture Review Board(ARB) deb nomlangan notijorat korxonasi tomonidan boshqariladi.

OpenMP ko`p oqimli ilovalarni tez va engil yaratishni Fortran va C/C++ algoritmik tillarda amalga oshiradi. OpenMP ning birinchi versiyasi 1997-yilda Fortran tili uchun yaratilgan. C/C++ dasturlash tillari uchun esa 1998-yilda yaratilgan. 2008-yilda esa OpenMP ning 3.0 versiyasi taqdim etildi.

OpenMP da parallel va ketma – ketlik [26]. Parallel muhitga kirilgandan so`ng yangi OMP_NUM_THREADS-1 oqimlar yaratiladi, har bir oqim o`zining unikal nomeriga ega bo`ladi, bunda dastlabki oqim 0 nomer bilan belgialanadi va u bosh oqim (master) bo`ladi. Qolgan oqimlar raqam sifatida butun sonlar 1 dan OMP_NUM_THREADS-1gacha bo`ladi. Oqimlar soni belgilangan parallel muhitda bajariladi va ushbu muhitdan chiqib ketishgacha o`zgarmay qoladi.

Parallel muhitdan chiqib ketgandan so`ng sinxronlash yordamida bosh oqimdan boshqa barcha oqimlar yo`q qilinadi.

Quyidagi misolda parallel direktivasi ishlashi keltirilgan. Natijada bosh oqim “1-ketma-ket muhit” matnini ekranga chop etadi, keyinchalik parallel direktivasi yangi oqimlarni hosil qiladi va ushbu oqimlarning har biri “parallel muhit” matnini ekranda chop etadi, keyin yaratilgan oqimlar tugatiladi va bosh oqim “2-ketma-ket muhit” matnini ekranga chop etadi.

```
#include "stdafx.h"
#include <omp.h>
using namespace System;
int main(array<System::String ^> ^args)
{
        Console::WriteLine(“1 – ketma – ket  muhit”);
#pragma omp parallel
{
    Console::WriteLine(“parallel  muhit”);
}
Console::WriteLine(“2 – ketma – ket  muhit”);
}
```

Ayrim hollarda tizimning o`zi parallel muhitda bajarilayotgan oqimlar sonini tizim resurslarini optimizatsiya qilish uchun dinamik ravishda o`zgartirishi mumkin.Oqimlar sonini dinamik ravishda o`zgartirish OMP_DYNAMIC o`zgaruvchisiga true qiymatni berish orqali amalga oshiriladi. Masalan, Linux operatsion tizimining bosh buyrug`i oynasida ushbu qiymatni quyidagi buyruq orqali amalga oshirilish mumkin:

```
export OMP_DYNAMIC q true;
```

Dinamik ravishda o`zgaradigan tizimlarda oqimlar soni odatda belgilanmagan bo`ladi va uning qiymati falsega teng bo`ladi.

omp_in_parallel() funksiyasi 1 qiymatni qaytaradi, agar aktiv holatdagi parallel muhitdan chaqirilgan bo`lsa.

Quyidagi misolda omp_in_parallel() funksiyasi qo`llanilgan. Funktsiyasi qaysi muhitdan chaqirilishiga qarab, “parallel muhit” yoki “ketma-ket muhit” qatorlarini chop etishda qo`llaniladi.

```
#include "stdafx.h"
#include <omp.h>
using namespace System;
void mode(void){
        if(omp_in_parallel())
                Console::WriteLine("parallel muhit");
        else
                Console::WriteLine("ketma-ket muhit");
}
int main(array<System::String ^> ^args){
        mode();
#pragma omp parallel     {
#pragma omp master               {
                        mode();
                }
        }
   return 0; }
```

C/C++ dasturlash tillarida [27] yuqoridagi barcha shartlar *single* direktivasi bilan birgalikda e`lon qilinadi.

Dasturning belgilangan qismini qaysi oqim bajarishi tavsiflanmaydi. Agarda nowait sharti e`lon qilinmasa, bitta oqim belgilangan fragmentni bajaradi, qolgan oqimlar uning ishini tugashini kutib turadi. single direktivasi umumiy o`zgaruvchilar bilan ishlaganda kerak.

Master direktivasikodning ma`lum bir qismini faqat bosh oqim bajarishi uchun belgilaydi. Qolgan oqimlar ushbu qismni o`tkazib yuborishadi va undan quyida turgan operator bilan dasturni ishlashini davom ettiradi. Ushbu direktivada

sinxronizatsiya amalga oshirilmaydi. C/C++ dasturlash tilida direktiva quyidagicha e`lon qilinadi:

#pragma omp master

Quyidagi misolda master direktivasining ishlashi keltirilgan. n o`zgaruvchi lokal hisoblanib, har bir oqim o`zining nusxalari bilan ishlaydi. Dastavval barcha oqimlar n o`zgaruvchiga 1 qiymatini o`zlashtirishadi. So`ngra bosh oqim n o`zgaruvchiga 2 qiymatini o`zlashtiradi va barcha oqimlar ushbu qiymatni ekranga chop etadi. Misolda ko`rinib turibdiki, masterdirektivasini har doim bitta oqim bajaradi. Ushbu misolda barcha oqimlar 1 sonini ekranga chiqarsa, bosh oqim dastlab 2 sonini, so`ngra esa 3 sonini ekranga chop etadi:

```
#include "stdafx.h"
#include <omp.h>
using namespace System;
int main(array<System::String ^> ^args)
{
	int n;
#pragma omp parallel private(n)
	{
		nq1;
#pragma omp master
		{
			nq2;
		}0
		Console::WriteLine("n ning birinchi qiymati: "Q n);
#pragma omp barrier
#pragma omp master
		{
			nq3;
		}
		Console::WriteLine("n ning keyingi qiymati: "Q n);
```

```
        }
    return 0;}
```

Quyidagi misolda private shartini ishlashi keltirilgan. Ushbu misolda parallel muhitda n o`zgaruvchi lokal o`zgaruvchi sifatida e`lon qilingan. Bu har bir oqimning n ning nusxalari bilan ishlashini bildiradi va har bir oqimning boshida n o`zgaruvchi initsializatsiya qilinadi. Dasturning bajarilish vaqtida n o`zgaruvchining qiymati to`rtta turli xil joylarda chop etiladi. Birinchi marta n o`zgaruvchining qiymati 1 ga o`zlashtirilgandan keyin ketma-ket muhitda chop etiladi, ikkinchi marta barcha oqimlar n o`zgaruvchining nusxasini parallel muhitning boshida chop etadi. Keyin barcha oqimlar o`zining tartib nomerini omp_get_thread_num() funksiyasi yordamida olingan qiymatini n ga o`zlashtirib chop etishadi. Parallel muhit tugagandan so`ng n o`zgaruvchining qiymati yana bir marta chop etiladi, bunda uning qiymati 1 ga teng bo`ladi (parallel muhit ishlashi davomida o`zgarmagan):

```
#include "stdafx.h"
#include <omp.h>
using namespace System;
int main(array<System::String ^> ^args)
{
        int n;
        Console::WriteLine("ketma-ket muhitga kirishdagi n ning qiymati: " Q n);
#pragma omp parallel private(n)
        {
Console::WriteLine("parallel muhitga kirishdagi n ning qiymati: " Q n);
                nqomp_get_num_threads();
Console::WriteLine("parallel muhitdan chiqishdagi n ning qiymati: " Q n);
        }
Console::WriteLine("ketma-ket muhitdan chiqishdagi n ning qiymati: " Q n);
    return 0;
```

}

C/C++ dasturlash tillarida dasturning parallel muhitda aniqlangan statik o`zgaruvchilar umumiy (shared) o`zgaruvchi hisoblanadi. Dinamik ajratilgan xotira ham umumiy hisoblanadi, ammo ko`rsatgich ham umumiy, ham lokal bo`lishi mumkin.

Threadprivate direktivasi C/C++ dasturlash tillarida umumiy bo`lgan o`zgaruvchilarni lokal ko`rinishiga o`tkazib berishi mumkin. Global obektlarni lokal o`zgaruvchilarini to`g`ri ishlatilishda dasturning turli qismlarida bir xil oqimlar tomonidan ishlatilishiga ishonch hosil qilish kerak. Lokal o`zgaruvchilarga turli parallel muhitlardan murojaat qilinsa, unda ularning qiymatini saqlab qolish uchun hajmli parallel muhitlar bo`lmasligi kerak, oqimlar soni ikkala muhitlarda ham bir xil bo`lishi kerak va OMP_DYNAMIC o`zgaruvchisining qiymati birinchi muhit boshlangandan ikkinchi muhit boshlangancha false deb o`rnatilgan bo`lishi kerak. Threadprivate tipida e`lon qilingan o`zgaruvchilar OpenMP direktivalarining copyin, copyprivate, schedule, num_threads_if shartlaridan boshqa shartlarida ishlatib bo`lmaydi.

Qulflar. OpenMP da sinxronizatsiya variantlaridan biri qulflar (locks) mexanizmidan foydalanish mumkin. Qulflar sifatida umumiy butun sonli o`zgaruvchilar (hajmi adresni saqlash uchun etarli bo`lishi kerak) ishlatiladi.Ushbu o`zgaruvchilar sinxronizatsiya primitivlari parametrlari kabi ishlatilishi kerak.

Qulflar quyidagi uchta holatda bo`lishi mumkin: initsializatsiya qilinmagan, bloklangan yoki bloklanmagan. Bloklanmagan qulf ayrim oqimlar tomonidan egallangan bo`lishi mumkin.Undan so`ng uning holatini bloklashga o`tadi.Bloklanmagan qulfni aynan o`sha oqim ozod qilishi mumkin, undan so`ng qulf bloklanmagan holatiga o`tadi.

Ikkita turdagi qulflar mavjud: oddiy qulflar va murakkab qulflar. Murakkab qulflar bitta oqim tomonidan ozod qilinishidan oldin ko`p marotaba egallanishi mumkin, oddiy qulflar esa faqat bir marta egallanishi mumkin. Murakkab qulflar uchun egallanganlik koeffitsienti (nesting count) tushunchasi kiritiladi. Dastlab

uning qiymati nolga teng bo`ladi, har bir egallanganda uning qiymati birga oshadi va har bir ozod qilinganda birga kamayadi. Murakkab qulflar egallanganlik koeffitsienti nolga teng bo`lsa bloklanmagan hisoblanadi.

Oddiy va murakkab qulflarni initsializatsiya uchun C/C++ dasturlash tillarida [12] quyidagi funksiyalar ishlatiladi:

void omp_init_lock(omp_lock_t *lock);

void omp_init_nest_lock(omp_nest_lock_t *lock);

C/C++ dasturlash tillarida [13] quyidagicha e`lon qilinadi:

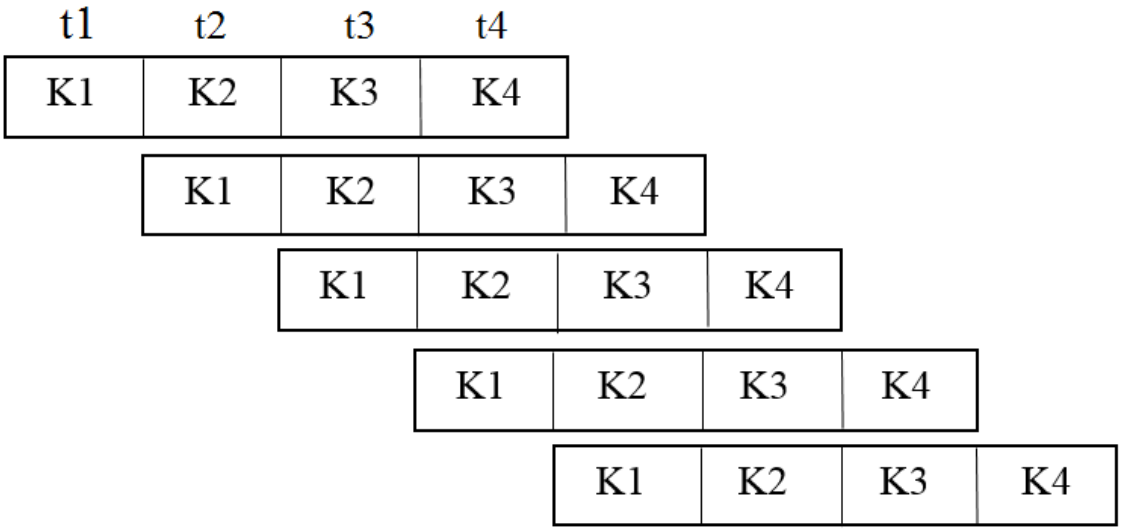
#pragma omp flush [(ro`yxat)]

Ushbu direktivani bajarish paytida oqimning registrlarida va kesh xotirasida saqlanayotgan hamma o`zgaruvchilarning qiymati asosiy xotiraga kiritiladi. Oqimning ishlashi davomida hamma o`zgaruvchilarning o`zgarishi qolgan oqimlar uchun ko`rinarli bo`ladi. Agarda qandaydir axborot chiqarish buferida saqlanayotgan bo`lsa, unda buferlar tashlab yuboriladi. Bunda operatsiya oqim tomonidan chaqirilgan ma`lumotlar bilan bajariladi, boshqa oqimlar tomonidan o`zgartirilgan ma`lumotlarga teginilmaydi [28].

1.2. Ko`p yadrolik protsessorlarda parallelashtirish amallarining tashkil etish tamoyillarini shakllantirish va protsessorlar uchun mo`ljallangan dasturiy paketlarda ishlash

Bir necha amallarni bir vaqtda bajarish g`oyasidan iborat bo`lgan ma`lumotlarni parallel xisoblash ikki xil ko`rinishi mavjud [29]. Bular: Parallel va konveyer.

Agar biror qurilma bitta amalni vaqt birligida bajarsa, u holda mingta amalni ming vaqt birligida bajaradi. Agar xuddi shunday bir vaqtda ishlay oladigan va bir–biriga mustaqil beshta qurilma mavjud deb qaralsa, u holda ular yuqoridagi mingta amalni mingta vaqt birligida emas, balki ikki yuzta vaqt birligida bajaradi. Xuddi shunday N ta qurilmadan iborat tizim 1000 ta amalni 1000/N vaqt birligida bajaradi. Unga o`xshash holatlarni hayotdan ham keltirish mumkin.



1.4-rasm. Konveyerli qayta ishlash arxitekturasi.

Konveyerli qayta ishlash [30]. Qo`zg`aluvchan vergulli shaklda tasvirlangan xaqiqiy ikkita sonni qo`shish uchun nima qilish kerak? Bunda bir qator mayda amallar bajariladi. Bular: tartibini solishtirish, tartibini tenglash, normallash va boshqa amallar. Dastlabki kompyuterlarning protsessorlari yuqorida keltirilgan barcha “mikro amallarni” har bir argumentlar juftligi uchun natijani xosil qilguncha ketma-ket bajargan va bundan keyin qo`shiluvchilarning keyingi juftligini qayta ishlashga o`tgan. Konveyerli qayta ishlash g`oyasida umumiy amal bir necha bosqichlarga ajratiladi. Har bir bosqich bajarilgandan keyin uning natijasi keyingi bosqichga beriladi va shu bilan birga kiruvchi ma`lumotlarning yangi qismi qabul qilinadi. Bunda oldin bajarilgan amallarni natijalarini qo`llash orqali qayta ishlash tezligi oshiriladi. Faraz qilaylik, amal beshta mikro amaldan iborat bo`lishi va ularni har biri bitta vaqt birligida bajaradi. Agar ajralmas yagona ketma-ket qurilma mavjud bo`lsa, u 100 ta argumentlar juftligini 500 vaqt birligida bajaradi. Agar har bir mikro amal konveyrli qurilmaning alohida bosqichida bajarilsa, u xolda bunday qurilmaning har bir qayta ishlash bosqichining beshinchi vaqt birligida birinchi 5ta argumentlari aniqlanadi. Birinchi natija vaqtning 5-birligidan keyin olinadi. 100 ta juftlikdan iborat to`plam esa 5+99=104 vaqt birligidan keyin olinadi. Ya`ni parallel qurilmaga nisbatan 5 marta tez bajariladi. Bir qarashda konveyerli qayta ishlashni parallel qurilmalarini o`rniga zarur miqdordagi konveyr qurilmalarini qo`llash

mumkindek ko`rinadi. Biroq bunda xosil bo`lgan tizim narxi va murakkabligi oshadi. Unumdorlik esa o`zgarmay qoladi.

Parallel dastur tuzish uchun, dasturdagi bir vaqtda va bir-biridan mustaqil protsessorlarda bajariladigan amallar guruhini ajratib olish kerak. Buning imkoniyati mavjudligi dasturda informattsion bog`liqliklar mavjudligi yoki yo`qligi bilan aniqlanadi [31]. Agar dasturning biror amali natijasi ikkinchi amal argumenti sifatida qo`llanilsa amallar informatsion bog`liq deb ataladi. Agar V amali A amaliga informatsion bog`liq bo`lsa, u holda V amali faqat A amali tugagandan keyin bajariladi. Agar A va V amallari informatsion bog`liqmas bo`lsa, u xolda algoritmda ularni bajarish ketma-ketligiga cheklanish qo`yilmaydi, xususan ular bir vaqtda bajarilishi mumkin. Shunday qilib, dasturni informatsion bog`liq amallarni aniqlashdan va ularni xisoblash qurilmalariga taqsimlashdan, sinxronlashdan va zarur kommunikatsiyani o`rnatishdan iborat bo`ladi.

Ma`lumotlarni almashish va qayta ishlash usullari. Massivli-parallel kompyuterlar paydo bo`lishi bilan parallel jarayonlarni aloqasini qo`llab-quvvatlovchi kutubxona interfeyslar keng tarqaldi. Bu yo`nalishning toifali vakiliga Message Passing Interface (MPI) interfeysi misol bo`ladi. Bu interfeys amalda vektor-konveyerli super-kompyuterdan tortib shaxsiy kompyutergacha bo`lgan barcha parallel platformalarda mavjud. Dasturning qaysi parallel jarayonlari dasturning qaysi qismida va jarayonlar bilan ma`lumotlar almashishi yoki o`z ishini sinxronlab borishi kerakligini dasturchining o`zi belgilaydi. Odatda parallel jarayonlarning manzil soxasi turlicha bo`ladi. Xususan, bu g`oyaga MPI da amal qilinadi. Boshqa texnologiyalarda, masalan Shmem da lokal (private) va umumiy (shared) o`zgaruvchilari qo`llaniladi. Bu o`zgaruvchilarga dasturning barcha jarayonlari murojat etishi mumkin va Put/Get toifasidagi operatsiyalar yordamida umumiy xotira bilan ishlash usuli tashkil etiladi. Linda sistemasi o`ziga xos xususiyatga ega bo`lib, unda ixtiyoriy ketma-ketlikda tilga to`rtta: in, out, read va eval funktsiyalarini qo`shadi va parallel dasturlar tuzish imkonini beradi. Afsuski, ushbu keltirilgan g`oya oddiy bo`lishiga qaramasdan, uni amalda qo`llash muammolar tug`diradi.

Xabarlarni almashish interfeys texnologiyasi. Taqsimlangan xotira parallel kompyuterlardagi keng tarqalgan dasturlash texnologiyasi MPI texnologiyasi [32] xisoblanadi. Bunday sistemalardagi parallel jarayonlarning o`zaro muloqat usuli bir-biri bilan xabarlar almashishdan iborat. Bu usul texnologiyasi nomi-Message Passing Interface (xabarlar almashish interfeysi) deb aks ettirilgan. MPI standartida sistema amal qilishi va dastur yaratishda foydalanuvchilar amal qilishi kerak bo`lgan qoidalar mavjud. MPI Fortran va Si bilan ishlashni qo`llab-quvvatlaydi. Interfeysning to`liq versiyasida 125 tadan ko`proq protsedura va funksiyalar mavjud. MPI MIMD(Multiple Instruction Multiple Data) stilidagi parallel dasturlarni qo`llab quvvatlaydi. Unda turli matnlar bilan berilgan jarayonlar birlashtiriladi. Biroq bunday dasturlarni tuzish va otladka etish murakkab. Shuning uchun amalda dasturchilar SPMD (SINGLE PROGRAM MULTIPLE DATA) parallel dasturlash modelidan foydalanishadi. Unda parallel jarayonlar uchun bitta dastur kodi qo`llaniladi.

MPI texnologiyasi[15]. Dasturni parallel qismini o`rnatish MPI_INIT(IERR). Qolgan MPI pratseduralar MPI_INIT chaqirilgandan keyin chaqirilishi mumkin. Xar bir dasturni parallel qismini o`rnatish faqat bir marta bajariladi. Si tilida MPI_INIT funksiyasida ko`rsatgichlar yordamida dasturning buyruq satridagi argc va argv parametrlari beriladi. Ular yordamida parallel jarayonlarga parametrlar beriladi.

Intel Parallel Studio da parallellashtirish funksiyasidan foydalanish [33]. Hozirgi kunga kelib klientli Windows ilovalarni ishlab chiqaruvchi mutaxassislarga mavjud yoki yangi yaratilayotgan ilovalar uchun kuchli va qulay bo`lgan, foydalanuvchi tizimlari multiyaderli protsessorlarning bazasida maksimal ravishda ishlay oladigan vositalar kerak bo`lmoqda.

Bugungi kunda eng keng tarqalgan Microsoft Visual Studio ilovalar ishlab chiqaruvchi mutaxassisning standart jamlanmasi hisoblanadi. Intel kompaniyasi Visual Studio ning imkomiyatlarini kengaytirish uchun, ya`ni Windows uchun yaratilayotdan parallel dasturlarni yengillashtirish va optimallashtirish uchun Visual Studio ga plug-in sifatida Intel Parallel Studio ni ishlab chiqdi. Hozirgi

kunga kelib dasturlarning unumdorligi ularning qanchalik parallellashgani va tizimdagi protsessorlarning oshishi bilan mutanosib bo`la olishi hisobga olinmoqda. Ideal dastur yaqin yillar ichida yadrolar soni ko`payib borayotgan vaqtda ishlab chiqarilayotgan yangi protsessorlarning hamma quvvatidan avtomatik tarzda foydalanadi.

Intel Parallel Studio-bu bir nechta vositalardan iborat bo`lgan jamlanma bo`lib, Microsoft Visual Studio ga bog`langan multiyaderli tizimlar uchun unumdorligi yuqori bo`lgan parallel dasturlar ishlab chiqaruvchi vosita hisoblanadi. Ushbu jamlanma Visual Studio ga plug-in bo`lishidan tashqari, dasturchi tomonidan ishlab chiqilayotgan dasturning hamma bosqichida jumladan, dasturning skeletidan boshlab oxirgi variantining optimizatsiyasigacha qatnashadi. Uning tarkibiga to`rtta alohida har biri ishlab chiqarish tsiklida qatnashadigan mahsulot kiradi.

Paket tarkibiga quyidagilar kiradi:

- intel parallel advisor-dasturning ishlab chiqarishning boshidan kodni parallellashtirish imkoniyatlarini topishga yordam beradi;
- intel parallel composer-parallel kod generatsiya qilish uchun mo`ljallangan bo`lib, dasturni kompilyator va ko`p oqimli algoritmlar uchun mo`ljallangan kutubxonalar majmusidan foydalaniladi;
- intel parallel inspector-parallel dasturni to`g`riligini tekshiradi va xotira bilan ishlash xatolarini topib beradi;
- intel parallel amplifier-dasturning multiyaderli platformalarda mutanosibligini oshirishga halaqit qiluvchi “tor joylari” ni aniqlaydi.

Yaratish tsikli uzluksiz va qaytariladigan hollarda ushbu vositalar bir muhitda uyg`unlik bilan va bir-biriga o`zaro bog`langan holda ishlashi juda muhim hisoblanadi [34]. Xatolar oqimini topish yoki dasturni optimallashtirish uchun alohida profilirovshiklarni ishga tushirish va boshqa eksperementlar qilish talab qilinmasligi uchun ular Microsoft Visual Studio muhiti bilan yaxlit birlashtirilgan. Ushbu vositalar Visual Studio ning toolbarida yoki menyusida joylashgan bo`lib, analiz natijalari esa proekt fayllari turgan joyda joylashtiriladi.

Visual Studio muhitida ushbu vositalarning qanchalik uyg`unlashganligini yozish qiyin masaladir.Ularning funksionalligi Visual Studioning davomi bo`lib, ajoyib parallel dasturlar tuzishda foydalanish muhim hisoblanadi.

Intel Parallel Advisor. Parallel dastur tuzishning ikki xil usuli mavjud.Birinchi usul mavjud dasturni ayrim izolyatsiyalangan qismlarini, odatda proyektning butun arxitekturasiga teginmaydigan algoritmlarni ishlash tezligini oshirish uchun qismli yoki butunlay parallellashtirishdir. Bunda dasturchi dasturni analiz qiladi va dasturda mikroprotsessorning maksimal resurslarini egallaydigan qismlarini belgilaydi. So`ngra proyektning strukturasi analiz qilinadi va algoritmni modifikatsiya qilish qarori qabul qilinadi. Parallel dastur tuzishning ikkinchi usuli parallel bajariladigan yuklanishlarni hisoblab borish hisoblanadi.Agar proyektni birgalikda bajarila oladigan qismlarga bo`lib bo`lsa, ularning dastur ko`rinishidagi realizatsiyasi odatda qiyin masala hisoblanadi.

Bu yerda yordamga Parallel Advistor keladi. Bu mutlaqo yangi vositalar jamlanmasi bo`lib, o`zida parallel dasturlarni noldan yaratish metodologiyasiga va ularni amalga oshirilishi uchun to`g`ri qoidalarga ega. Parallel Advistor dasturlar parallel yaratilganda effektiv bo`lmagan holatlarini topadi va kerakli yechimlarni beradi. Bundan tashqari parallel kutubxonalarni ishlatish bo`yicha barcha bilimlar simpllar va shablonlar ko`rinishida yig`ilgan bo`lib, boshlang`ich bosqichda ulardan foydalanish uchun maksimal ravishda yengillashtirilgan. Tayyor dasturni parallellashtirish uchun, ilovaning to`g`riligini tekshirish va optimizatsiya qilish uchun Advistor “yo`l ko`rsatuvchi oqim” ni yoki workflow ni taqdim qiladi [35].

Intel Parallel Composer-IPP unumdorlik va TBB parallel kutubxonalari bilan Visual Studioga bog`langan bo`lib, parallel dasturlashni endigina boshlaganlar va o`zining dasturini unumdorligini Intel texnologiyasi bo`yicha oshiruvchilar uchun qulay hisoblanadi.

Composer quyidagi komponentalardan tuzilgan:

- IPP kutubxonasida funksiya sifatida yaratilgan hisoblash primitivlari intel platformalaridagi algoritmlarini yuqori unumdorligini kafolatlaydi;

- oldingi versiyalarda bo`lmagan, lekin OpenMP 3.0 standartning yangi versiyalarida mavjud bo`lgan multitasking dan foydalanadi;
- vektorli operatsiyalarni bajaruvchi ma`lumotlarning yangi tipi Valarray kodni bir muncha engillashtiradi, kompilyator esa unumdorlikni oshirish uchun SIMD-instruksiyalarni ishlatib, samarador binar kodni generatsiya qiladi;
- kompilyator tomonidan C++ ning standart elementlarini qo`llab-quvvatlaydi.

Intel Parallel Inspector-ushbu vosita bugungi kunda eng ko`p talab qilingan va kutilgan vosita hisoblanadi, u ko`p oqimli dasturning verifikatsiya bosqichida bajarilish turg`unligini oshirib xatoliklardan qutilish uchun yordam beradi. Inspector faqatgina testirovshik komandalar (QA team) bilan ishlatilmaydi.

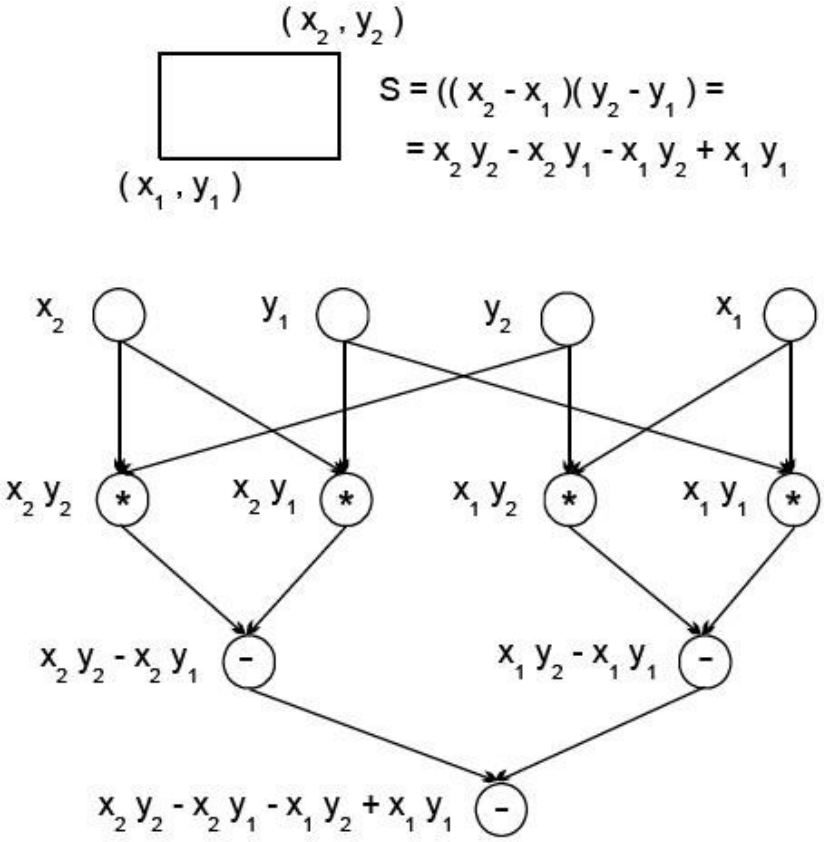
Parallel Inspector quyidagicha ishlaydi. Ikkta turdagi xatoliklar klassini topadi va ular bir-biridan alohida ishga tushiriladi: ko`p oqimli xatoliklar va xotira bilan ishlash xatoliklari. Xatoliklarning oxirgi klassi dasturchilar uchun yaxshi ma`lum bo`lib, xotiraning noto`g`ri ishlatilishi va stekning butunligini buzilganligini yoki mavjud bo`lmagan manzillardan foydalanishni aniqlovchi turli vositalardan foydalanishgan. Xatoliklarning ikkinchi klassi ko`p oqimli dasturlarning tabiati bilan bog`liq [36]. Ular parallel dasturlarni yaratish paytida hosil bo`ladi va ularni aniqlash qiyin hisoblanadi.

Intel Parallel Amplifier-unumdorlik profilirovshiki bo`lib, ilova tomonidan multiprotsessorli platforma qanchalik samarador ishlatilishini va dasturning unumdorligini oshirishga halaqit beruvchi nozik joylarni aniqlaydi.

1.3. Parallel algoritmni amalga oshirilishini ta`riflash va ma`lumot almashish usullari va algoritmning bajarilishi uchun sarflanadigan vaqtni hisoblash

Algoritmning orasida tanlangan hisoblash sxemasi doirasida yo`l bo`lmagan operatsiyalari parallel bajarilishi mumkin (1.5-rasmdagi hisoblash sxemasi uchun masalan, parallel tarzda oldin ko`paytirishning barcha operatsiyalari, keyin ayirishning birinchi ikki operatsiyasi bajarilishi mumkin). Algoritmni parallel

bajarilishini ta`riflashning ehtimoli bo`lgan usuli quyidagidan iborat bo`lishi mumkin.



1.5-rasm. Graf ko`rinishidagi algoritmning hisoblash modeliga misol.

p algoritmni bajarish uchun foydalanilayotgan protsessorlar soni bo`lsin. Shunda hisoblarni parallel bajarish uchun har bir $i \in V$ operatsiya uchun $P_i$ protsessor operatsiyasini bajarishi uchun foydalaniladigan raqam va $t_i$ operatsiyani bajarilishini boshlash vaqti ko`rsatiladigan to`plam (jadval) berilishi kerak.

$$H_p = \{(i, P_i, t_i) : i \in V \qquad (1.1)$$

Jadval amalga oshadigan bo`lishi uchun $H_p$ to`plam berilganda quyidagi talablar bajarilishi kerak:

1. $\forall i,\ j \in V: t_i = t_j \Rightarrow P_i \neq P_j$, `ni, bir protsessorning o`zi vaqtning bir momentida turli operatsiyalarga yo`naltirilishi kerak,

2.$\forall i, j \in R \Rightarrow t_j \geq t_i + 1$, ya`ni, operatsiya bajarilishining belgilangan vaqtiga barcha zarur ma`lumotlar hisoblab bo`linishi kerak.

G algoritmning H_p jadval bilan hisoblash sxemasi p protsessorlardan foydalanilib bajariladigan $A_p(G, H_p)$ algoritmning parallel modeli sifatida ko`rib chiqilishi mumkin. Parallel algoritmning bajarilish vaqti jadvalda foydalaniladigan vaqtning maksimal qiymati bilan aniqlanadi.

$$T_p(G, H_p) = max_{i \in V} t_i + 1 \qquad (1.2)$$

Hisoblarning tanlangan sxemasi uchun algoritm bajarilishining minimal vaqtini ta`minlovchi jadvaldan foydalanish maqsadga muvofiq:

$$T_p(G) = min_{H_p} T_p(G, H_p) \qquad (1.3)$$

Bajarilish vaqtining kamaytirilishi eng yaxshi hisoblash sxemasini tanlash yo`li bilan ham ta`minlanishi mumkin:

$$T_p(G) = min_G T_p(G) \qquad (1.4)$$

$T_p(G, H_p)$, $T_p(G)$va T_p baholar parallel algoritmni bajarish vaqtining ko`rsatkichlari sifatida ishlatilishi mumkin. Bundan tashqari, ehtimoli bo`lgan maksimal parallellikni tahlil qilish uchun algoritmning eng tez bajarilish bahosini aniqlash mumkin:

$$T_\infty = min_{p \geq 1} T_p \qquad (1.5)$$

T_∞ bahoni protsessorlarning cheklanmagan sonidan foydalanishda parallel algoritmni bajarishning ehtimoli bo`lgan minimal vaqti sifatida ko`rib chiqish

mumkin (odatda parallel kompyuter deb ataladigan, protsessorlarning cheksiz soniga ega hisoblash tizimining konsepsiyasi parallel hisoblarni teoretik tahlilida keng ishlatiladi).

T_1 baho bir protsessordan foydalanishda algoritmning bajarilish vaqtini aniqlaydi va shu bilan vazifani echish algoritmining so`ngi varianti bajarilish vaqtini bildiradi. Bunday bahoning qurilishi parallel algoritmlarni tahlil qilishda muhim vazifa hisoblanadi, chunki u parallellikdan foydalanish effektini aniqlash uchun zarur (vazifani echish vaqtini tezlashtirish).

Ayonki,

$$T_1(G) = |\bar{V}|, \qquad (1.6)$$

bu erda $|\bar{V}|$, eslatib o`tamiz, kiritish cho`qqilarisiz G hisoblash sxemasining cho`qqilari soni. Agar T_1 bahoni aniqlashda vazifani echishning faqat bir tanlangan algoritmi bilan cheklanilsa va quyidagi kattalikdan foydalanilsa,

$$T_1 = min_G T_1(G) \qquad (1.7)$$

Unda bunday bahodan foydalanishda olinadigan izlanish ko`rsatkichlari tanlangan algoritmning parallellashtirilish effektini bildirishini aytib o`tish zarur. O`rganilayotgan matematik hisoblash vazifasining parallel echilishi effektivligini aniqlash uchun ketma-ket echish vaqtini turli ketma-ket algoritmlarni hisobga olgan holda aniqlash kerak, ya`ni quyidagi kattalikdan foydalanish,

$$T_1^* = minT_1 \qquad (1.8)$$

Bu erda minimum operatsiya iushbu vazifani echishning ehtimoli bo`lgan barcha ketma-ket algoritmlarining to`plami bo`yicha olinadi.

Parallel algoritmni bajarish vaqtini baholash xususiyatlarini bildiruvchi isbotsiz teoriyalarni keltiramiz.

1-teorema. Parallel algoritmni bajarishning ehtimoli bo`lgan minimal vaqti algoritm hisoblash sxemasining maksimal yo`li uzunligi bilan aniqlanadi, ya`ni

$$T_\infty(G) = d(G) \qquad (1.9)$$

2-teorema. Algoritmning hisoblash sxemasida qandaydir chiqish cho`qqisi uchun kirishning har bir cho`qqisidan yo`l mavjud bo`lsin. Bundan tashqari, sxema cho`qqilarining kiruvchi darajasi (kiruvchi yoylar soni) 2 dan oshmasin. Shunda parallel algoritmni bajarishning ehtimoli bo`lgan minimal vaqti pastda quyidagi qiymat bilan cheklangan:

$$T_\infty(G) = log_2 n \qquad (1.10)$$

bu erda n algoritm sxemasida kirish cho`qqilari soni.

3-teorema. Foydalanilayotgan protsessorlar sonini kamaytirishda algoritmni bajarish vaqti protsessorlar soni kamayishi kattaligiga proporsional kattalashadi, ya`ni

$$\forall q = cp, 0 < c < 1 \implies T_p \leq cT_q \qquad (1.11)$$

4-teorema. Ishlatilayotgan protsessorlarning har qanday soni uchun parallel algoritmni bajarish vaqti uchun quyidagi yuqori baho adolatli:

$$\forall p \implies T_p < T_\infty + \frac{T_1}{p} \qquad (1.12)$$

5-teorema. T_∞ ehtimoli bo`lgan minimal vaqt bilan taqqoslanadigan algoritmni bajarish vaqtlariga protsessorlarning $p \sim \frac{T_1}{T_\infty}$ tartib sonida erishish mumkin, aynan,

$$p \geq T_1 \,/\, T_\infty \implies T_p \leq 2T_\infty \qquad (1.13)$$

Protsessorlarning kam sonida algoritmni bajarish vaqti 2 martadan oshishi mumkin emas, protsessorlarning mavjud sonida hisoblarning eng yaxshi vaqti, ya`ni

$$p < T_1 \,/\, T_\infty \implies \frac{T_1}{p} \leq T_p \leq 2\frac{T_1}{p} \qquad (1.14)$$

Keltirilgan tasdiqlar parallel algoritmlarni shakllantirish qoidalari bo`yicha quyidagi tavsiyalarni berish imkonini beradi:

1. Algoritmning hisoblash sxemasini tanlashda ehtimoli bo`lgan minimal diametrli grafdan foydalanish kerak (1-teorema);

2. parallel bajarish uchun protsessorlarning maqsadga muvofiq soni $p \sim T_1 \,/\, T_\infty$ kattalik bilan aniqlanadi (5-teorema);

3. parallel algoritmni bajarish vaqti 4 va 5 teoremalarda keltirilgan kattaliklar bilan yuqoridan cheklanadi [37].

1.4. Parallel algoritmlarning samaradorlik ko`rsatkichlari

p protsessorlar uchun parallel algoritmdan foydalanishda olinadigan tezlanish hisoblarni bajarishning ketma-ket varianti bilan taqqoslaganda quyidagi kattalik bilan aniqlanadi:

$$S_p(n) = T_1(n) \,/\, T_p(n) \qquad (1.15)$$

ya`ni skalyar EHMda vazifani echish vaqtining parallel algoritmni bajarilishiga ketgan vaqtga nisbati sifatida (n kattalik echiladigan vazifaning hisoblash murakkabligini parametrlash uchun ishlatiladi va masalan, vazifaning kiruvchi ma`lumotlarining soni sifatida tushinilishi mumkin).

Vazifalarni echishda protsessorlarni parallel algoritm bilan ishlatish samaradorligi quyidagi nisbat bilan aniqlanadi:

$$E_p(n) = T_1(n) \big/ \Big(pT_p(n) \Big) = S_p(n) \,/\, p \qquad (1.16)$$

(samaradorlik kattaligi protsessorlar vazifani echish uchun haqiqatdan ham ishlatiladigan algoritm bajarilish vaqtining o`rta hissasini aniqlaydi).

Keltirilgan nisbatlardan eng yaxshi holatda $S_p(n) = p$ va

$E_p(n) = 1$ ekanligini ko`rsatish mumkin. Parallel hisoblarning effektivligini baholash uchun ushbu ko`rsatkichlarni amaliy qo`llanilishida quyidagi ikki muhim jihatni hisobga olish kerak:

Ma`lum sharoitlarda tezlanish foydalanilayotgan jarayonlar sonidan katta bo`lishi mumkin $S_p(n) = p$ bu holatda yuqori chiziqli tezlanishning mavjudligi haqida gapirishadi. Bunday holatlarning paradoksligiga qaramasdan (tezlanish

protsessorlar sonidan oshadi), amaliyotda yuqori chiziqli tezlanish o`z o`rniga ega. Bunday hodisaning sabablaridan biri ketma-ket va parallel dasturlarni bajarishning noravonligi bo`lishi mumkin. Masalan, bir protsessorda vazifalarni echishda barcha qayta ishlanadigan ma`lumotlarni saqlash uchun operativ xotira etishmovchiligi bo`ladi, va, buning natijasida, ancha sekin tashqi xotiradan foydalanish zarur bo`ladi (bir nechta protsessorlardan foydalanish holatida esa operativ xotira protsessorlar o`rtasida ma`lumotlarni taqsimlanishi hisobiga etarli bo`lishi mumkin). Yuqori chiziqli tezlanishning yana bir sababi qayta ishlanadigan ma`lumotlar hajmiga qarab vazifalar echilishi murakkabligi bog`liqligining chiziqsiz xarakteri bo`lishi mumkin [38]. Masalan, pufakli saralashning mashhur algoritmi zarur operatsiyalar sonining tartiblanadigan ma`lumotlar soniga kvadrat bog`liqligi bilan xarakterlanadi. Buning natijasida, protsessorlar o`rtasida saralanadigan massivni taqsimlashda protsessorlar sonidan oshadigan tezlanish olinishi mumkin. Yuqori chiziqli tezlanishning manbai ketma-ket va parallel metodlarning hisoblash sxemalarining farqi ham bo`lishi mumkin.

Diqqat bilan ko`rib chiqilganda bir ko`rsatkich bo`yicha parallel hisoblar sifatini oshirishga urinishlar (tezlanish yoki samaradorlik) boshqa ko`rsatkich bo`yicha holatning yomonlashuviga olib kelishiga e`tibor qilish mumkin, chunki parallel hisoblarning sifat ko`rsatkichi qarama-qarshi hisobalanadi. Masalan, tezlanishning oshirilishi odatda protsessorlar sonini oshirish hisobiga ta`minlanishi mumkin, bu, qoidaga ko`ra, effektivlik pasayishiga olib keladi. Va aksincha, effektivlikning oshirilishiga ko`p holatlarda protsessorlar sonini kamaytirishda erishiladi.

(eng ideal samaradorlik $E_P\,(n) = 1$ bir protsessordan foydalanilganda oson ta`minlanadi).Buning natija sifatida parallel hisoblar metodlarini ishlab chiqish ko`pincha tezlanish va samaradorlikning ma`qul ko`rsatkichlarini hisobga olgan holda qandaydir kelishilgan tanlovni ko`zda tutadi [39].

Keyinchalik kiritilgan tushunchalarni ko`rsatish uchun navbatdagi punktda sonli qiymatlarning ketma-ketligi uchun xususiy summalarni hisoblash vazifasining echilishiga o`quv misoli ko`rib chiqiladi. Bundan tashqari, ushbu

ko`rsatkichlar hisoblash matematikasining shunday vazifalarini echishda keyinchalik ko`rib chiqiladigan barcha parallel algoritmlarning samaradorligini xarakterlash uchun ishlatiladi.

1.5 Masalaning qo'yilishi. Ushbu bitiruv malakaviy ishida mavjud ma'lumotlarni qayta ishlovchi algoritmlarni tahlil qilish, tasvirlarni qayta ishlovchi parallel algoritmlarning samaradorlik ko'rsatkichlarini baholash asosiy masala qilib olingan. Mazkur ishda bir va ko'p yadroli protsessorlar faoliyati va ishlash tezligi ham o'rganilishi kerak bo'ladi.

II. BOB. MA`LUMOTLARNI QAYTA ISHLASHDA QO`LLANILGAN PARALLEL ALGORITMLARI VA UNUMDORLIK NATIJALARINING TAHLILI

2.1. Tasvirlarni Veyvlet-jarayon orqali qayta ishlashda parallellashtirish algoritmlarining unumdorlik darajasi

Bu bobda ma`lumotlarni qayta ishlashda parallel metodlarining dasturiy algoritmlarda qo`llanilishini va ularning amaliyotda ko`p yadroli protsessorlarda tadbiqi, dasturiy til sifatida C++ va Visual Studio [40] tilining qo`llanishi va parallelashtirishni ko`p oqimli qilib tashkillashtirib beradigan OpenMP va OpenCV komplyatorlari direktiva imkoniyatlari to`g`risiga bag`ishlanadi. Bundan tashqari bu bo`limda dasturda qo`llaniladigan parallel algoritmlarining ko`p yadroli protsessorlarda qo`llash natijasida erishilgan unumdorlik darajasining, xotiradan o`qish va yozish amallarini optimallashtirish maqsadida tasvirlarni siqish muammolarini yechishda dasturiy vositalardan va oqimlarga ajratish usullaridan foydalanish haqida aytib o`tiladi.

Bu muammolarni yechish maqsadida siqish amallarida Veyvlet-jarayon va Gauss usullari orqali qayta ishlashni va amallar ketma ketligini bajarishda paralellashtirish algoritmlaridan foydalanildi. Veyvlet-jarayon va Gauss usullari orqali tasvir ma`lumotlarini siqish – hozirgi paytda keng qo`llaniladigan siqish usullaridan biri hisoblanadi. Sodda holda tushuntirish maqsadida tasvirning tashkil topish qismlari haqida qisqacha to`xtalib o`tiladi.

Tasvir — ikki o`lchamli matritsadan tashkil topgan ma`lumotlar to`plami bo`lib, matritsaning har bir yacheykasi yana uchta qiymatni o`zida saqlagan massivdan iborat [41]. Agar biz oddiy oq-qora tasvirlarni oladigan bo`lsak, unda har bir yacheykada rang o`rnida qism bo`lakning yorqinligi haqidagi ma`lumot saqlanadi, ya`ni 0 va 1. Bu holatda 0 qora rang haqida ma`lumot beradi, 1 qiymati esa oq rang haqida ma`lumot beradi. Tasvirning sifat darajasi va xotiradan egallaydigan xajmi matritsada ishtirok etadigan qiymatlarning 0 dan 255 gacha bo`lgan sonlarning qatnashish darajasi va tasvirning yorqinlik, aniqlik va tiniqlik

qiymat darajasiga bog`liq bo`ladi. Ammo tasvir o`lchami va uning kengaytmasi tasvir sifat darajasiga va xajmining oshib yoki kamayib ketishiga ham ta`sir ko`rsatadi. Buning natijasida xotiradan me`yordan tashqari ko`p joy egallashi yoki sifatining pasayib ketishiga sabab bo`ladi.

Hattoki uncha katta bo`lmagan tasvir ham xotiradan ko`p joy egallashi mumkin. Agar biz tasvirning har bir piksel yorqinligini bir baytdan qiymatlasak, unda oddiygina FullHD (1920×1080) formatining bir kadri taxminan xotiradan ikki megabayni egallaydi. Bunda umumiy ketma-ketlikdan tashkil topgan multimedia axboroti xotiradan juda katta joy egallaydi. Natijada multimedia ma`lumotlarini xotiraga joylashtirish, saqlash va uni o`qish kabi amallarda ayrim turdagi muammolarni kelib chiqishiga va tizim ishiga salbiy ta`sir ko`rsatishiga olib keladi.

Xotirani samaradorligini oshirish uchun:

- Veyvlet-jarayon yordamida siqish usullari qo`llaniladi;
- Tasvirni bo`lmasdan turib tasvir zarrachalarini optimal xajmini qidirib topiladi ;
- tasvirni umumiy holatini 2^n kesimlarga bo`linadi har bir kesim aloxida qayta ishlanadi.

Shuning maqsadida tasvirlarni siqish amallarining optimal yo`llari yaratiladi. Tasvirlarni qayta ishlaganda shunday amallarni optimallashtirish zarurki, bunda qayta ishlash natijasida hosil bo`lgan tasvir ma`lumotlari xotiradan kam joy egallasin. Tasvir foydalanuvchi uchun kompyuter ekranida akslanganda esa yana sifat darajasini tiklagan holda namoyish etishi muhim hsoblanadi [20].

Hozirgi davrda siqish yo`llarining bir qancha usullari mavjud hisoblanadi. Ularning nomlarini tasvir kengaytmasidan bilish mumkin. Bular o`zining yutuq tomonlari bo`lgani bilan kamchilik tomonlari ham mavjud. Shu sababga ko`ra ularni qo`llanilish yo`nalishlari mavjud.

Misol sifatida, oddiy real xayotdagi tasvirni PNG kengaytmada saqlasak, xotirada egallagan xajmi bizni xayratda qoldirishi mumkin. Bunga asosiy sabab

shundaki, har bir tasvir bo`lagi yorqinligi qo`shni qism yorqinligi bilan kamdan kam holda bir xil qiymatda bo`ladi. Bu esa siqish jarayonida yaxshi natija bermaydi.

Asosan siqish algoritmlari siqiladigan ma`lumotlarda qandaydir qonuniylik mavjud bo`lganda yaxshi natija beradi. Misol uchun tasvir ma`lumotlari ichida 0 lar soni 100 ni tashkil etadi deylik, bunda tasvirni xotirada saqlashda faqatgina 100 sonini yozib qo`yish kifoya. Tasvirni xotiradan o`qish davomida dekodirlaydigan dasturlar 100 sonini o`qigan «0» lar ketma ketligi deb tushunadi. Ayrim holatlarda 0 lar ketma ketligi o`rtasida 1 soni mavjud bo`lsa unda siqish natija bermasligi mumkin. Chunki bu sonlar ketma ketligini 2 ta qiymat bilan ifodalashga to`g`ri keladi. Ifodalar sonining oshishi har bir o`zgaruvchi uchun qo`shimcha xotiradan joyni talab qiladi va xotiradan o`qish vaqtida o`zgargan qiymat uchun protsessor navbatiga yana qo`shimcha amallar ketma ketligini yuklashga to`g`ri keladi. Bu esa amal bajarish qonuniga asosan qo`shimcha vaqt talab qiladi [42].

Lekin yana bir savol tug`iladi: nega aynan tasvirning har bir detalini saqlashimiz kerak? Inson tasvirga qaraganda unda nima aks etganini anglay oladi, undagi yorug`likning tebranish darajasini ilg`ay ham olmaydi, shundan kelib chiqib tasvirdagi ayrim ko`z ilg`amas qiymatlarni boshqa son bilan ifodalasa ham bo`ladi. Shunda tasvir ma`lumotlarini siqqanimizda yaxshiroq natijaga erishamiz.

Yuqorida keltirib o`tilgan nazariy ma`lumotlarga asosan tasvirlarni Veyvlet jarayon orqali siqishda parallellashtirish algoritmlaridan foydalanildi. Algoritmlarning amaliyoti sifatida Microsoft Visual Studio 2010 dasturiy paketida Visual C++ (v100) [23,26,30] va Intel C++ Compiler XE 11.0 kompilyatoridan foydalangan holda dastur tuzildi. C++ kutubxonalaridan foydalanildi, shuningdek ko`p oqimlilikni tashkillashtirish sifatida OpenMP kompilyator direktivasidan foydalanildi [43].

Dasturda ketma ket va oqimlarga ajratish natijalari va ular orasida erishilgan unumdorlik darajasini ko`rish uchun 4 ta natijalar oynasi keltirilgan. Bundan tashqari dasturni ishga tushirgach qayta ishlanashi kerak bo`lgan tasvirni yuklash uchun tugma qo`yilgan. Quyidagi rasmda dasturning asosiy oynasi keltirilgan.

Dasturni ishga tushurish uchun Microsoft Visual C++ 2010 dasturiy majmuasining bir necha paketlarini o`rnatish zarur bo`ladi. Bu paketlar quyidagi bibliotekalarni: C Runtime (CRT), Standard C++, ATL, MFC, OpenMP va MSDIA ni ishlashini ta`minlaydi.

Dasturga tasvir yuklangach, uni siqin tugmasiga bosish orqali Veyvlet jarayoni amalga oshiriladi va tasvir ma`lumotlarini siqish amalga oshiriladi. Amallar ketma ketligi quyidagilardan iborat:

- dasturga yuklangan tasvirning RGB – qiymatlarini baytli massivga yuklab olinadi;
- RGB – qiymatlarni kvantlangan umumiy ranglar kompanentalarini YCrCb ga kodlanadi;
- veyvlet jarayoni amalga oshiriladi;
- ko`p o`lchamli massivlar qatorini bir o`lchamli massivga o`zlashtirib olinadi;
- ixtiyoriy siqish algoritmi orqali hosil bo`lgan ma`lumotlar qatorini siqib olinadi.

Yuqoridagi bosqichlarning birinchi bosqichida biz tasvir o`lchamlari standart bo`lmagan xolat uchun tashkil etildi. Tasvir dasturga yuklangach uning o`lchamlari Visual S++ dasturlash tilida mavjud funksiyalar orqali o`zlashtirib olinadi.

Bitmap^ bmp q gcnew Bitmap(pictureBox1->Image);

BitmapData^ bmpData q bmp->LockBits(Rectangle(Point(), bmp->Size), ImageLockMode::ReadOnly, PixelFormat::Format24bppRgb);

int width q bmp->Width; (tasvir enining o`lchami)

int height q bmp->Height; (tasvir bo`yining o`lchami)

O`zlashtirib olingan qiymatlardan kelib chiqan holda dinamik xotira hosil qilinadi. Dinamik xotira hosil qilishdan asosiy maqsad shundaki, tasvir ma`lumotlari qayta ishlash davomida o`z qiymatlarini o`zgartiradi va ma`lumotlar xajmiga teng xotiradan joy ajratiladi:

unsigned char**** b q new unsigned char***[mat];

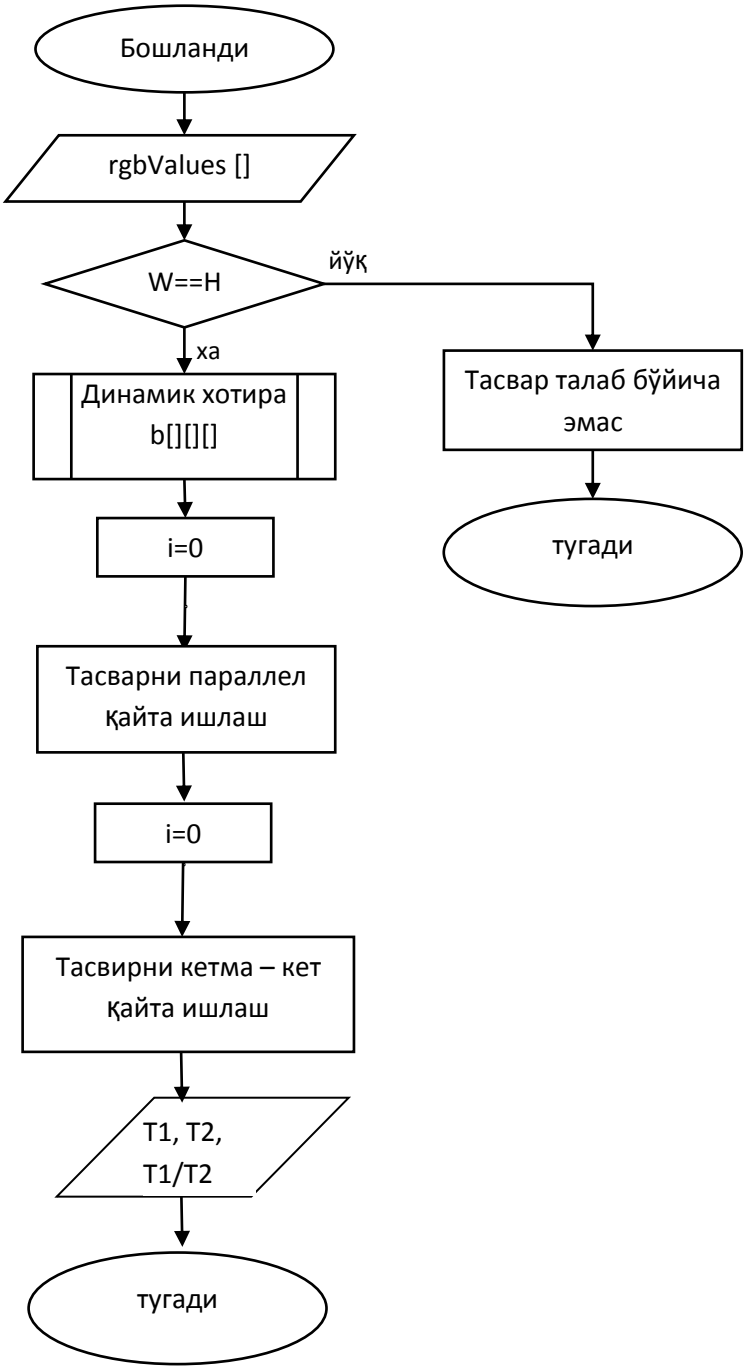
for(int k q 0; k < mat; kQQ)

```
{
        b[k] q new unsigned char**[3];
        for (int i q 0; i < 3; iQQ)
        {
                b[k][i] q new unsigned char*[width];
                for (int j q 0; j < width; jQQ)
                {
                b[k][i][j] q new unsigned char[height];
                        }
                }
        }
```

Endi esa dasturning ishlash tartibi bilan tanishib o`tamiz.

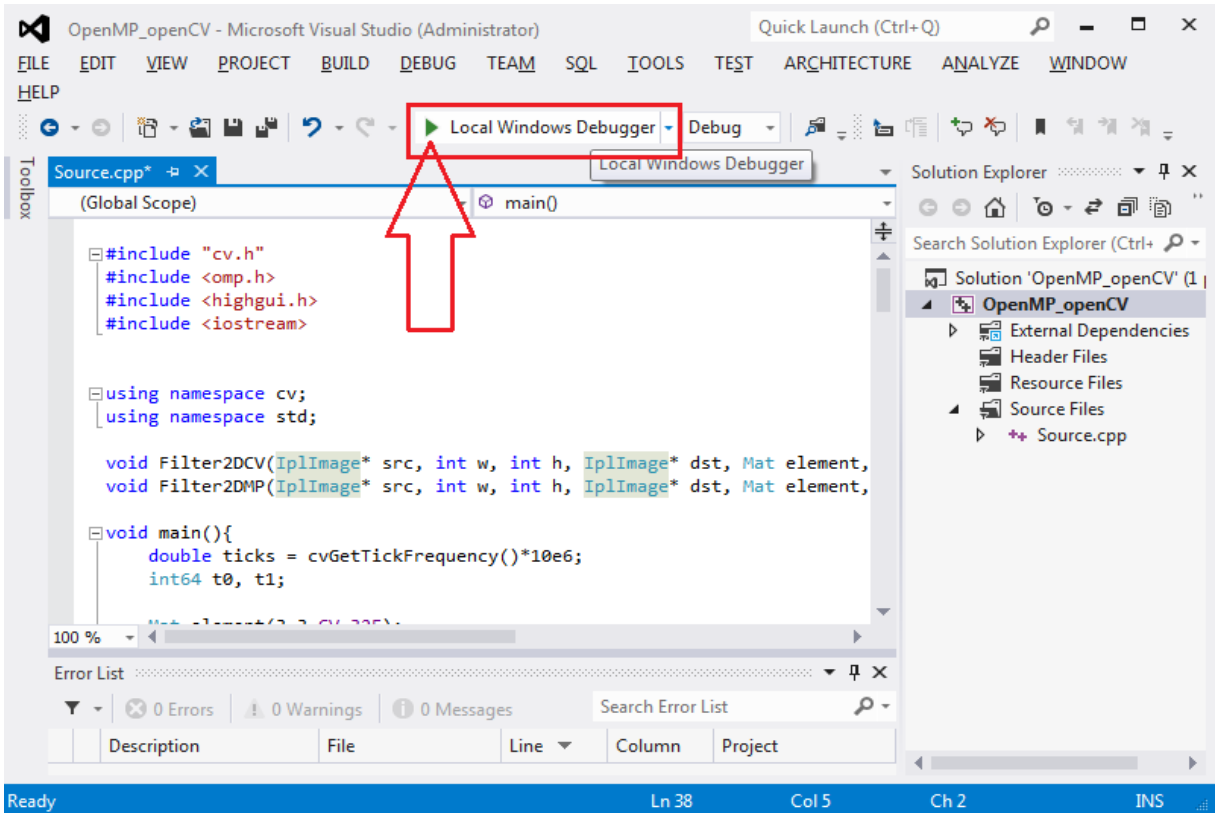
Veyvlet jarayonni amalga oshirganimizda an`anaviy Jpeg algoritmidan foydalanildi. Bunda tasvir matritsasini boshlang`ich o`lchami sifatida 8x8 deb olindi.

Dasturning algoritm qismi quyidagi ko`rinishga ega:

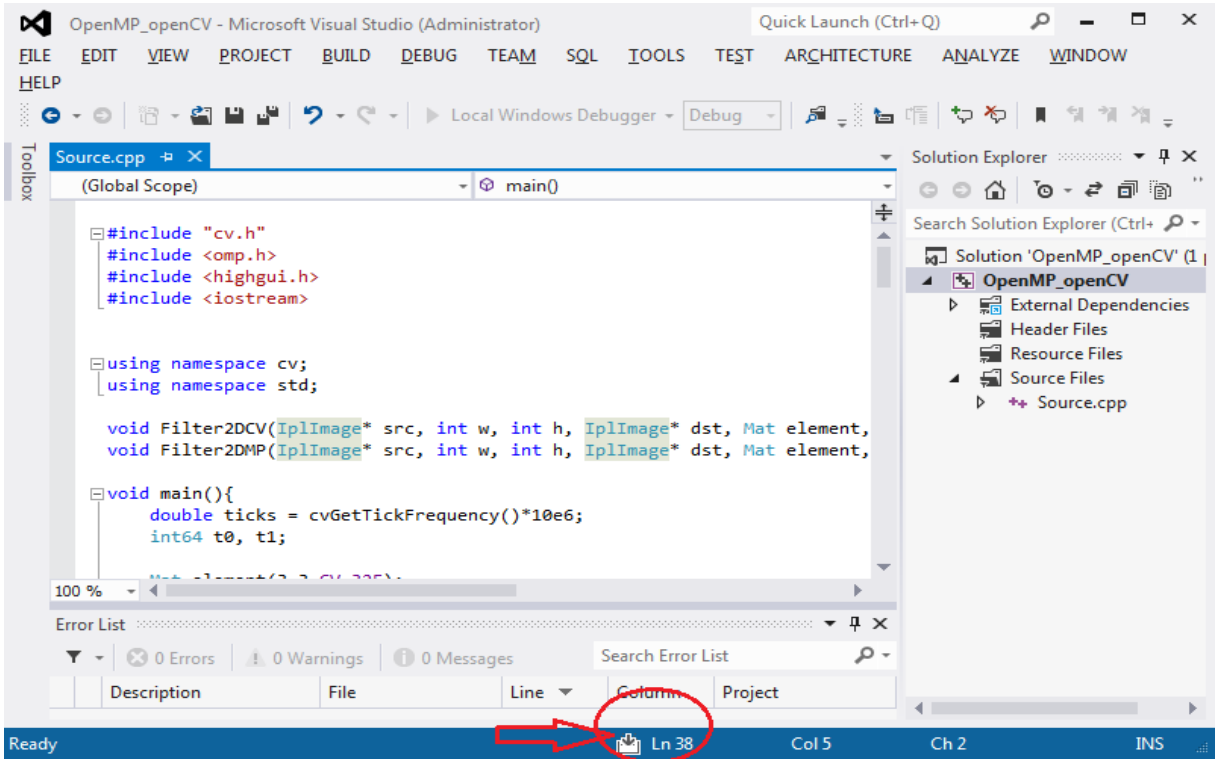


2.1-rasm. Ma`lumot tahlilini amalga oshiruvchi dasturining algoritmi.

Microsoft visual studioning OpenMP va OpenCV kompilyatori yordamida parallel algoritmlardan foydalanib ma`lumotlarni filtrlashni amalga oshiramiz (2.1-rasm).



2.2-rasm. Microsoft visual studio ni ishchi oynasi.



2.3-rasm. Microsoft visual studio ni OpenMP va Open CV kompilyatori.

2.4-rasm. Kiruvchi tasvir.

Ushbu keltirilgan tasvirni berilgan algoritm asosida filtrlaymiz:

2.5-rasm. Oddiy algoritm asosida filtrlangan tasvir.

Hosil bo`lgan tasvirni parallel algoritm asosida filtrdan o`tkazamiz:

2.6-rasm. Parallel algoritm asosida filtrlangan tasvir.

Yuqoridagi tasvirning filtrlash jarayoni uchun ketgan vaqtni ko`rib chiqamiz:

2.7-rasm. Filtrlash jarayoni uchun ketgan vaqt.

2.2. Bir va ko`p yadroli protsessorlarda bajariladigan amallarni oqimlarga ajratish algoritmlari

Tasvir ma`lumotlarini veyvlet jarayon orqali qayta ishlaganimizda bir – biriga bog`liq bo`lmagan jarayonlar ketma ketligini protsessorga yuklanishini

ko`rishimiz mumkin. Bunda ayrim hollarda bir xil amallar ketma ketligi navbat bilan qayta ishlanilishi vaqtdan yo`qotilishga olib keldi. O`rtacha holatda 2048x2048 o`lchamga ega rangli tasvirni qayta ishlaganimizda 2 yadroli protsessorda taxminan 1251,653 mls va 4 yadroli protsessorda esa 387,37 mls vaqt ketishini ko`rishimiz mumkin. Bu balkim uncha katta bo`lmagan qiymatdir, ammo multimedia tizimlarida ma`lumotlar ketma ketligidan tashkil topgan animatsiyani qayta ishlaganda sezilarli darajada kutib qolishlarni guvohi bo`lishimiz mumkin [44].

Hozirgi vaqtgacha ishlab chiqarilgan dasturiy vositalar asosan bir yadro protsessorlar uchun tadbiq etilgan edi. Axborot texnologiyalarining to`xtovsiz ravishda rivojlanib borayotgani, an`anaviy algoritmlarning unumdorlik darajasining nisbatan pasayishini ko`rsatmoqda. Bunga asosiy sabab axborot xajmining keskin oshishi va ma`lumotlarning tuzilish jixatidan murakkablashib borayotganligidadir [45]. Hozirga qadar yaratilgan dasturiy vositalar faqatgina ma`lum belgilangan chuklanishlar evaziga amallar ketma ketligini bajarishar edi, bu qandaydir shartlarni qanoatlantira olardi. Ayniqsa multimedia tizimlarida axborotga ishlov berish murakkab jarayonlardan tashkil topadi. Oddiygina tasvir ustida bajariladigan qayta o`zgartirish amallarini oladigan bo`lsak, avvolo xotira masalasiga duch kelamiz. Tasvir sifatining yaxshilanib borishi va tasvir ma`nosining murakkabligi uning xajmiga ham ta`sir etmasdan qolmaydi [46]. Bunda xotiradan tasvir ma`lumotlarini o`qish va qayta ishlangan natijalarni yozish kabi amallar ish ko`lamining asosiy vaqtini sarflab qo`yadi. Albatta protsessorda amalga oshadigan jarayonlar yuqoridagi muammolardan qolishmaydi. Xotiradan protsessorga yuklash jarayonini amalga oshirishda shinaning bo`shash vaqtini poylash va band qilish vaqti taqsimlash xam tizim tomonidan cheklanganligi tasvir usida bajarilishi kerak bo`lgan ishni sekinlashishiga olib keladigan sabablardan biridir. Ayniqsa protsessor bilan bog`liq tomonlari ham muhim ham murakkab jarayon hisoblanadi [26].

Protsessorlarda ko`p yadrolik tushunchasining paydo bo`lishi «ko`p oqimlilik» tushunchasini ham dasturiy ta`minotda ko`proq tilga olinishga sabab

bo`ldi. Bu ikki tushuncha hozirgi axborot texnologiyalarda uzviy bog`liq «juftlik» hisoblanadi.

Yuqorida keltirilgan dasturda ham tasvirlar ustida qayta ishlash amallarini oqimlarga ajratishning ikki xil usuli qo`llanilgan. Har bir usulda ham OpenMP ning oqimlarga ajratish direktivalari qo`llanilgan. Bu usullarni novbat bilan tahlil qilib chiqamiz [47].

Birinchi usulda tasvirdan olingan ma`lumotlar massivini veyvlet jarayon orqali qayta ishlaganimizda bir – biriga bog`liq bo`lmagan amallar ketma – ketliklarni ajratib olamiz va shu amallarni oqimlarga ajratish direktivalarga birlashtiramiz. Izoh sifatida shuni takidlash lozim: agar parallelashtirilayotgan amallar ketma – ketliklar umumiy bog`liq o`zgaruvchiga bog`langan bo`lsa, oqimlarga ajratish jarayonida dastur xatoliklari paydo bo`lishi mumkin. Misol sifatida dasturda qo`llanilgan tasvirning RGB – qiymatlari baytli massivga o`zlshtirish jarayonini ko`rib chiqsak:

#pragma omp parallel for shared(rgbValues) G`G` parallel jarayonning boshlanishi

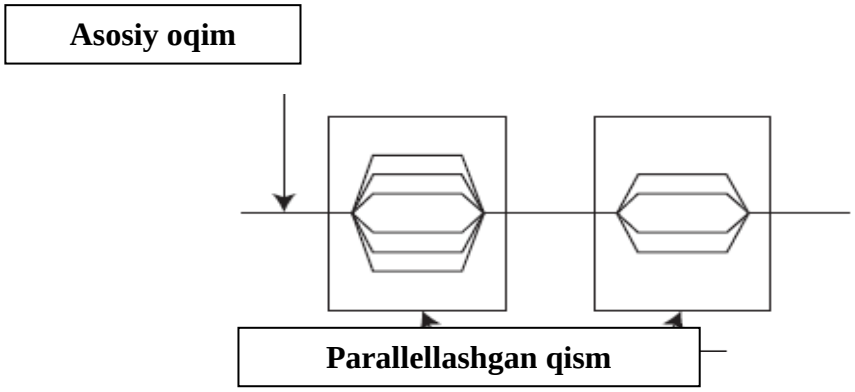
```
for(int k = 0; k < mat; k++)
{
 if( k % N == 0 ) { s1q0; s2++; }
 for (int i = 0; i < width; i++)
{
        for (int j = 0; j < height; j++)
        {
b[k][0][i][j] = rgbValues[j + (s1 * width) * width * 3 + i + (s2 * width) * 3 + 0];
b[k][1][i][j] = rgbValues[j + (s1 * width) * width * 3 + i + (s2 * width) * 3 + 1];
b[k][2][i][j] = rgbValues[j + (s1 * width) * width * 3 + i + (s2 * width) * 3 + 2];
}
```

```
        }s1++ ;
    }
    #pragma omp barrier // parallelashtirish jarayoni sinxron holda yakunlanadi
    }
```

OpenMPda sinxronlash. Keltirilgan kodda s1 va s2 o`zgaruvchilar rgbValues massiv uchun umumiy o`zgaruvsi bo`lib qolgan, bu esa kodning boshlanishida e`lon qilingan #pragma omp parallel for direktivaning shared(rgbValues) shartini qanoatlantirmaydi. Bu holatni nazarda tutgan holda shartni qanoatlantiruvchi qo`shimcha o`zgaruvchilar massivini qo`shish va tsikl ichidan umumiy bo`lgan o`zgaruvchilarni olib tashlash kifoya. Buning natijasida o`zgartirish kiritilgan kodlar ketma ketligi quyidagi ko`rinishda bo`ladi:

```
    int s1[] q { 0,1,0,1 };
    int s2[] q { 1,1,2,2 };
    #pragma omp parallel for shared(rgbValues) // parallel jarayonning boshlanishi
        for(int k = 0; k < mat; k++)
        {
         for (int i = 0; i < width; i++)
        {
            for (int j = 0; j < height; j++)
            {
    b[k][0][i][j] = rgbValues[j + (s1[k] * width) * width * 3 + i + (s2[k] * width) * 3 + 0];
    b[k][1][i][j] = rgbValues[j + (s1[k] * width) * width * 3 + i + (s2[k] * width) * 3 + 1];
    b[k][2][i][j] = rgbValues[j + (s1[k] * width) * width * 3 + i + (s2[k] * width) * 3 + 2];
    }
        }
    #pragma omp barrier // parallelashtirish jarayoni sinxron holda yakunlanadi.
```

Ixtiyoriy ketma-ket yoki parallel dastur ikki xil qismlar to`plamidan iborat bo`ladi: ketma-ket qism va parallel qism. Ketma-ket bajariladigan qismda faqat bitta bosh oqim (master) yaratiladi. Ushbu oqimda dastur initsializatsiya qilinadi, shuningdek, dasturning yakunlanishi ham ushbu oqim orqali bo`ladi. Ketma -ket bajariladigan dasturda parallellashtirish qismi uchraganda faqat bitta bosh oqim yaratiladi va ushbu oqim dasturning bajarilishi davomidagi yagona oqim bo`lib qoladi. Parallel bajariladigan dasturda parallellashtirish qismi uchraganda bir qancha parallel oqimlar (slave) yaratiladi. Yaratilgan parallel oqimlar turli xil protsessorli yoki bitta protsessorli hisoblash tizimlarida bajarilishi mumkin [48]. Bitta protsessorli hisoblash tizimlarida parallel oqimlar protsessordan joy olish uchun o`zlari o`rtasida raqobatlashadi.Raqobatlashishniboshqarishoperatsion tizimning rejalashtiruvchisi tomonidan maxsus algoritmlar yordamida amalga oshiriladi. Linux operatsion tizimidagi vazifalarni rejalashtiruvchi protsesslarni qayta ishlashda standart hisoblangan arg`umchoq algoritmi (round-robin)dan foydalanadi. Bunda faqat tizim administratorlari ushbu algoritmi tizim vositalari yordamida o`zgartirishi mumkin. Parallel dasturning prinsipial sxemasi 3.6-rasmda keltirilgan:



2.8-rasm . Parallel dasturning sxemasi.

Parallel dasturning ishga tushirilishi natijasida bosh oqimning initsializatsiya qismidan va uning protsessi bajarilishidan boshlanadi. Unda zaruratga qarab parallel oqimlar yaratiladi va ularga kerakli ma`lumotlar uzatilib bajariladi. Parallel oqimlar dasturning bitta parallel qismida bir-biridan mustaqil ravishda yoki bir-biri

bilan aloqa o`rnatgan holda bajarilishi mumkin. Bir-biri bilan aloqa o`rnatgan holda bajarilishi dasturning ishlab chiqarilishini murakkablashtirishi mumkin [49].

Bunday hollarda dasturchi parallel oqimlar o`rtasida ma`lumotlar uzatilishini rejalashtirishi, tashkil etishi va sinxronizatsiyalashi kerak bo`ladi. Parallel dasturlarni ishlab chiqishda parallellashtirish qismlaridagi parallel oqimlarni mustaqil ravishda ishlashini ta`minlash maqsadga muvofiq bo`ladi [50]. Parallel oqimlar o`rtasida ma`lumotlar almashishi uchun OpenMP da umumiy o`zgaruvchilar ishlatiladi. Umumiy o`zgaruvchilarga turli parallel oqimlardan murojaat qilinganda ma`lumotlarga ega bo`lishda konflikt holati yuzaga kelishi mumkin.Ushbu konfliktni oldini olish uchun sinxronizatsiya (synchronization) protsedurasidan foydalanish mumkin. Sinxronizatsiya protsedurasini bajarishga juda ko`p vaqt ketishini hisobga olish kerak va iloji bo`lsa ushbu protsedurani kam hollarda ishlatish maqsadga muvofiq bo`lar edi. Buning uchun dastur tuzishda ma`lumotlar strukturasini juda puxta o`ylashga to`gri keladi.

OpenMP da ma`lumotlar modeli. OpenMP da ma`lumotlar modeli hamma oqimlar xotira muhiti uchun umumiy va har bir oqim uchun lokal xotira qismi mavjud deb taxmin qilinadi [8,11,12,15]. OpenMP da parallel muhitdagi o`zgaruvchilar 2 turga bo`linadi:

- shared (umumiy, hamma oqimlar ushbu turdagi o`zgaruvchilarni ko`radi);
- private (lokal, har bir oqim o`zgaruvchining nusxasini o`zida ko`radi).

Umumiy o`zgaruvchi hamma qismlar uchun har doim faqat bitta nusxada bo`ladi va barcha oqimlarga bitta nomda bo`ladi.Lokal o`zgaruvchilar e`lon qilinganda, har bir oqim uchun bir xil tipdagi va o`lchamdagi nusxalari yaratiladi. Bitta oqimdagi lokal o`zgaruvchining qiymati o`zgarsa ham qolgan oqimlardagi nusxalariniki o`zgarmaydi.

Bundan tashqari dasturning kod qismida har bir kvadrat matritsa ustida parallelashtirish amallari qo`llanilgan. For tsiklida oqimlarga ajratish #pragma omp parallel for diriktivasi bilan amalga oshiriladi. Qo`yidagi dasturiy kodda uning qo`llanilishi keltirilgan:

#pragma omp parallel for

```
for (j = 0; j < n; j++)
{
xWavelet[j] = ImgArray[Component][dwPos][j];
}
```

Parallel direktivasi yordamida parallel muhit hosil qilinadi. C/C++ dasturlash tilida [22,25] quyidagicha ko`rinishda bo`ladi:

#pragma omp parallel [shart [[,] shart]…]

Yuqorida keltirilganlarga asosan quyidagicha xulosa qilish mumkin: dasturni parallel qismlarga ajratilishi va parallel protsesslarni ishlab chiqish muhimdir.

2.3. Ko`p yadroli protsessorlarda erishilgan teskorlikni tahlil qilish

Multimedia tizimlarida tasvirlarni qayta yo`llaganda veylet jarayonlarini tadbiq etish yaxshi natija beradi. Ayniqsa parallelashtirish algoritmlaridan foydalanish unumdorlik darajasini oshirishga yordam beradi. Tasvirlarni qayta ishlaganda birinchi usul yordamida amalga oshiramiz. Bu usulning amalga oshish algoritmi quyidagicha amalga oshadi: dasturga yuklangan tasvir 2^N qiymat bilan amalga oshadi. N ning qiymati tasvir bo`lingan matritsasi 16x16 o`lchamga ega bo`lguncha amalga oshadi. Bu holatda misol sifatida N = 1 ga teng bo`lganda tasvir 4 ta matritsaga ajralada (2.8-rasm) . Bundan ko`rinadi algoritm OpenMP dan foydalanganda 2 yadroli protsessorda amalga oshirilgada 2 ta oqimga 2 marta bo`lib beriladi va tsikl 2 marta aylanishga to`g`ri keladi. Ketma ket amalga oshirganda esa bu amallar bajarilganda tsikl 4 marotaba aylanishga to`g`ri keladi. Bu holatni nazariy jixatdan taxlil qiladigan bo`lsak, unumdorlik 2 marotaba oshadi degan xulosaga kelishimiz mumkin [51]. Ammo OpenMP yordamida oqimlarga ajratganda xotira va protsessor bilan oqimlarni tashkillashtirganda ma`lum vaqt sarflanadi. Chunki oqim yaratilganda xotiraga hosil bo`layotgan oqim uchun dinamik xotira yaratish lozim va oqim o`z jarayonini yakunlaganda dinamik xotirani o`chirish kerak bo`ladi. Keyingi jarayon bu oqimlarni protsessorda bajarilish uchun navbatga qo`yish [52]. Bu holda ham ma`lum darajada vaqt

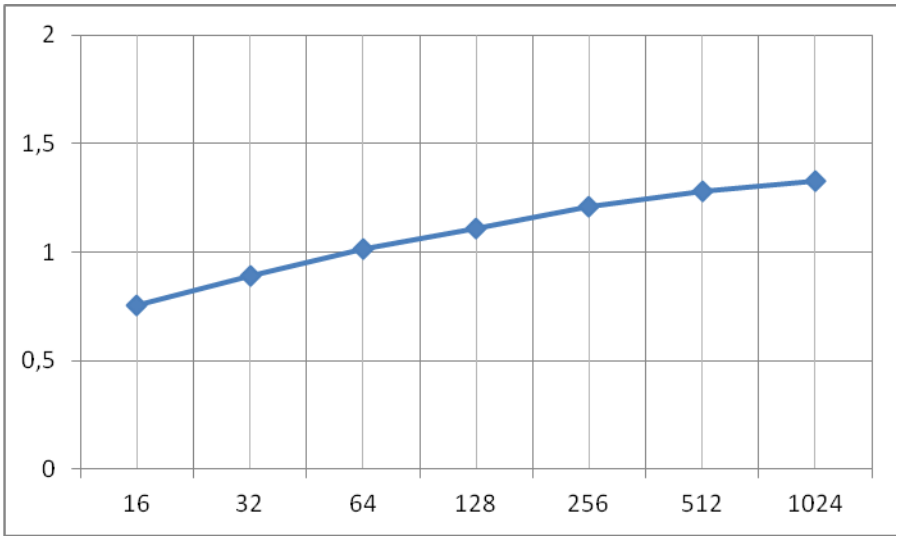
sarflanadi. OpenMP paketidan foydalangan holda tasvirni qayta ishlaganda 2 yadroli protsessorda qayta ishlaganda quyidagi 1 – jadvalda ko`rsatilgan natijalarga erishildi. Jadvalning birinchi ustunida qayta ishlanayotgan tasvirning nechta matritsaga bo`linishi va bo`lingan har bir matritsaning qanday o`lchamga ega ekanligi ko`rsatilgan.

2.1 – jadval.

2 yadroli protsessorda tasvirlarni qayta ishlaganda sarflangan vaqt va unumdorlik.

N x N	**Oddiy algoritm yordamida**	**OpenMP yordamida**	**OpenCV yordamida**	**Unumdorlik**
4 ta 1024	5141,619	3862,97	3781,4	1,331
16 ta 512	5381,641	4230,043	4123,03	1,279
64 ta 256	5498,038	4550,084	4435,01	1,208
256 ta 128	5657,647	5166,368	4987,1	1,109
1024 ta 64	5954,337	5862,022	5621,02	1,016
4096 ta 32	7029,633	7888,272	7678,2	0,891
16384 ta 16	9878,041	13121,636	12234,5	0,753

Natijalardan shuni ko`rishimiz mumkinki, bir oqimli ketma ket qayta ishlash amalga oshirganda OpenMP va OpenCV yordamida amalga oshirganga nisbatan ko`proq vaqt talab qilishi mumkin.



2.9-rasm. OpenMP yordamida 2 yadroli protsessorda erishilgan unumdorlikni grafik ko`rinishi.

OpenMP yordamida tasvirni qayta ishlaganda *#pragma omp parallel* diriktivasi protsessorni maksimal holda oqimlarga ajratishga harakat qiladi. Ya`ni 2 yadroli protsessorda maksimal holda 2 ta oqimni yaratib berishi mumkin. Ammo oqimlarni maksimal holda belgilab berish bilan unumdorlikka erishib bo`lmaydi [53]. Chunki oqimlarni tashkillashtirishga ham bog`liq hisoblanadi. Bu vazifani OpenMP kompilyatori tashkillashtirib beradi. Unumdorlik natijasini 2.9 – rasmda ko`rishimiz mumkin.

Aynan shu holatni 4 yadroli protsessorda amalga oshirganimizda 2–jadvaldagi natijalarni ko`rishimiz mumkin.

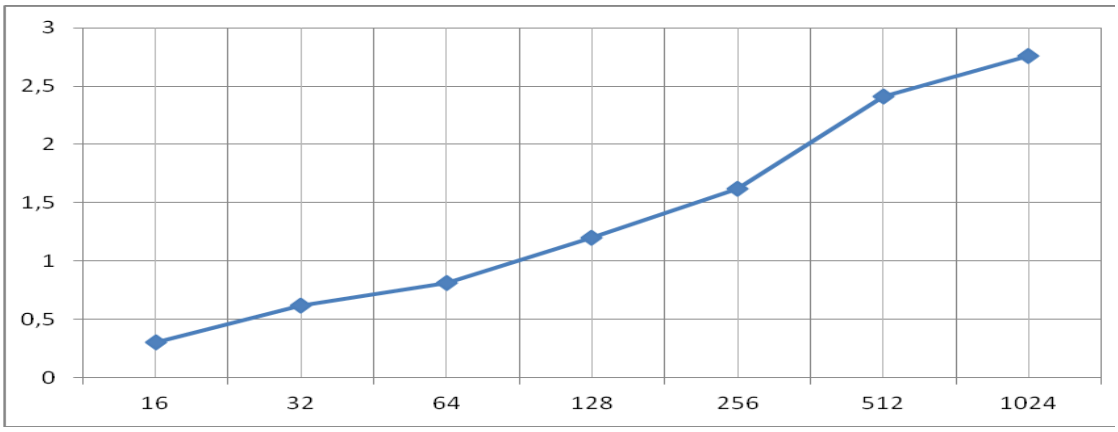
Yadrolar sonining oshishi oqimlar sonini oshishiga imkon beradi. 4 yadroli protsessorda maksimal holda 4 ta oqim tashkillashtirishimiz mumkin.

2.2 – jadval.

4 yadroli protsessorda tasvirlarni qayta ishlaganda sarflangan vaqt va unumdorlik

N x N	**Odatda**	**OpenMP**	**OpenCV**	**Unumdorlik**
4 ta 1024	1873,058	679,337	658,211	2,757
16 ta 512	1693,052	701,696	689,234	2,413

64 ta 256	158 1,531	977, 97	954,42	1,617
256 ta 128	201 7,086	1676 ,533	1579,2 41	1,203
1024 ta 64	144 6,008	1776 ,122	1721,2 01	0,814
4096 ta 32	163 0,208	2636 ,226	2564,1 25	0,618
16384 ta 16	214 3,645	7029 ,12	6875,0 1	0,305

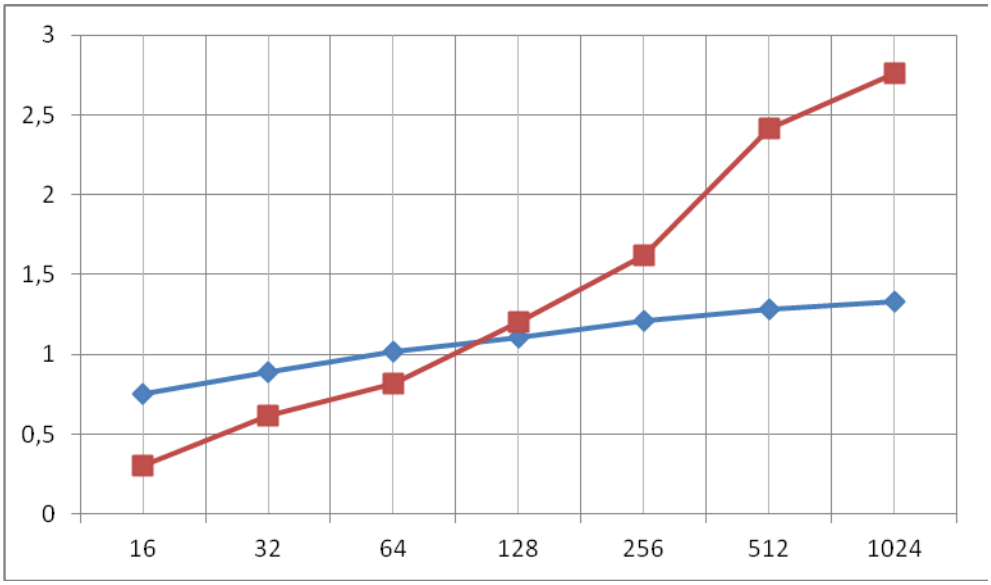


2.9-rasm. OpenMP yordamida 4 yadroli protsessorda erishilgan unumdorlikni grafik ko`rinishi.

Unumdorlik natijasini 2.9-rasmdagi grafikda shuni ko`rishimiz mumkinki, tasvir bo`lingan kesimlar o`lchami 64 dan kichkina bo`lsa algoritm unumdorlik bermaydi. Unumdorlikni optimal qiymatlar deb 512-1024 oralig`i aniqlandi.

Yuqoridagi natijalardan shuni ko`rishimiz mumkinki, unumdorlik natijasi *n* yadroli protsessorlarda *n* dan oshmasligini ko`rishimiz mumkin. 2 va 4 yadroli protsessorlarni unumdorligini solishtirganimizda quyidagi 3.10-rasmda farqini ko`rishimiz mumkin.

Tasvirlarni qayta ishlashda unumdorlikni oshirishning keyingi usulimizda algoritm quyidagicha amalga oshadi: tasvirni qayta ishlaganda avvoli tasvir pikselini minimal holda 64 ta qiymatini qayta ishlashni boshlaydi va maksimal holda tasvirninig o`lchamigacha oboradi.



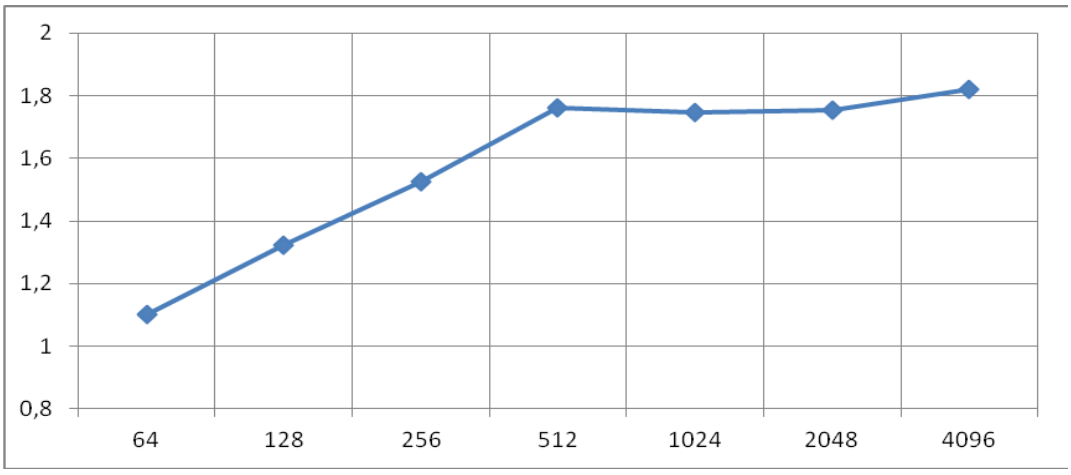
2.10-rasm. OpenMP yordamida 4va 2 yadroli protsessorda erishilgan unumdorlikni grafik ko`rinishi(qizil-4 yadro, ko`k-2 yadro)

Tasvirni kesmalarga bo`lmasdan zarralarni qayta ishlanganda ularning boshlang`ich o`lchami sifatida 64 deb olinda va 2^n bo`yicha davom ettirildi (2.3-jadval).

2.3 – jadval. 2 yadroli protsessorda tasvirlarni **nxn** o`lchamda qayta ishlagandagi unumdorlik.***n x n***	**Unumdorlik**
64	1,101
128	1,324
256	1,524
512	1,761
1024	1,747
2048	1,755
4096	1,819

Bu usulni 2 yadroli protsessorda qo`llaganimizda 3-jadvaldagi natijalarga erishildi.

nxn qiymatining o`zgarishini quyida keltirilgan grafikda yaqol ko`rishimiz mumkin.



2.11-rasm. Tasvirni nxn o`lchamdagi zarrachalarga bo`lib qayta ishlaganda 2 yadroli protsessorda erishilgan unumdorlik

Grafikdan shunday xulosa qilish mumkinki, tasvirning *nxn* zarralar sonini oshirganimiz sari unumdorlik darajasi oshib borar ekan. Optimal yechim sifatida 512 qiymati tanlandi, chunki tajriba natijalardan 512 qiymatida protsessor unumdorligi yaqqol namoyon bo`ldi.

2.4 – jadval.

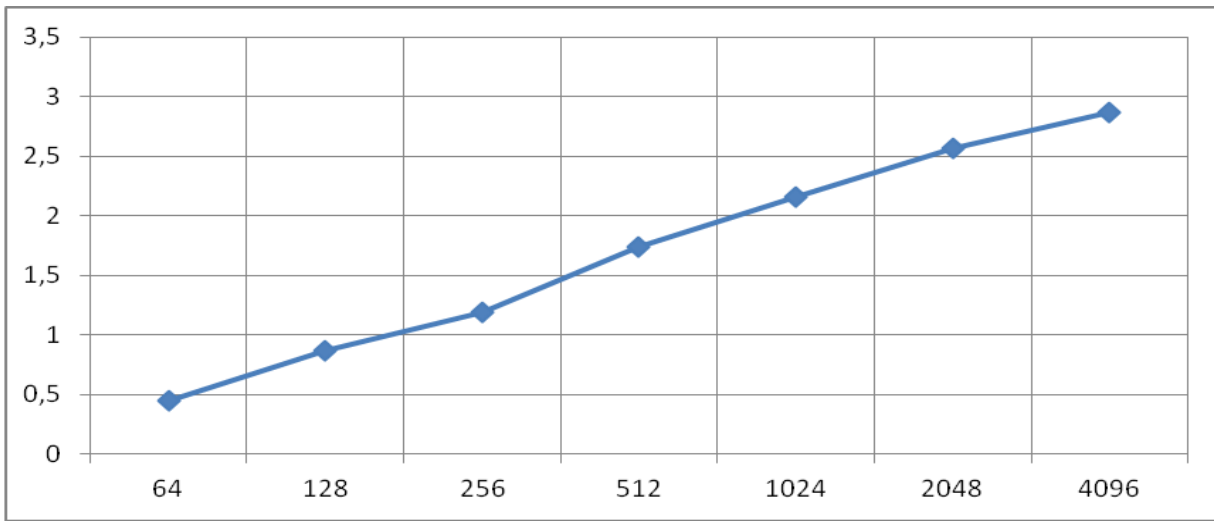
4 yadroli protsussorda tasvirlarni nxn o`lchamda qayta ishlaganda erishilgan unumdorlik natijalar

n x n	**Unumdorlik**
64	0,452
128	0,864
256	1,189
512	1,743
1024	2,157
2048	2,569

4096	2,869

Endi esa aynan shu usulni 4 yadroli protsessorda amalga oshirib ko`rdik. Uning unumdorlik natijasi 4-jadvalda berilgan.

Bunda ham natijani grafik ko`rinishida namoyish qilganimizda yuqoridagi ko`rinishga yaqin grafik paydo bo`ldi:



2.12-rasm. tasvirni **nxn** o`lchamdagi zarralarga bo`lgan holda qayta ishlaganda 4 yadroli protsessorda erishilgan unumdorlik .

2.12-rasmdagi grafik ko`rinishidan 4 yadroli protsessorlarda tasvirni zarralarga bo`lib qayta ishlaganimizda unumdorlik darajasi keskin o`sib borish holitini ko`rishimiz mumkin. Ammo grafikda maksimal holda tasvir zarra o`lchami 4096 ga tengligini ko`rishimiz mumkin. Buninig sababi protsessorning kesh xotirasining xajmiga bog`liqdir. Chunki tasvir zarralarini protsessorda qayta ishlaganimizda kesh xotiradan vaqtinchalik dinamik xotira yaratish lozim bo`ladi. Bu esa tasvir ma`lumotlarining xajm jixatdan kattaligi xotira etishmovchiliklarini keltirib chiqaradi.

Klaster hisoblash tizimlari uchun kommunikatsion sohani qurishning keng qo`llaniladigan usullaridan biri klasterning protsessor tugunlarini yagona hisoblash tarmog`iga birlashtirish uchun konsentratorlar (hub) yoki o`tkazuvchilardan

(switch) foydalanish hisoblanadi. Bu holatlarda klaster tarmog`ining topologiyasi to`liq grafni o`zida namoyon qiladi biroq kommunikatsion operatsiyalarni bajarishning bir hilligida ma`lum cheklanishlar mavjud [54]. Shunday qilib, konsentratorlardan foydalanilganda vaqtning har bir kechayotgan momentida ma`lumotlar yuborilishi faqat ikki protsessor tuguni o`rtasida bajarilishi mumkin.O`tkazuvchilar protsessorlarning bir nechta kesishmaydigan juftlarining o`zaro ta`sirlashuvini ta`minlashi mumkin.

Klasterlarni yaratishda boshqa, ko`p qo`llaniladigan echim kommunikatsion operatsiyalarni bajarishni asosiy usuli sifatida paketlarni yuborish metodidan foydalanishdan iborat (qoidaga ko`ra, TCP_IP protokol asosida amalga oshiriladi).

Keyinchalik tahlil qilish uchun ushbu tarqalgan tip klasterlarini tanlagan holda (to`liq graf ko`rinishidagi topologiya, xabarlarni yuborishning paket usuli), ikki protsessor tugunlari o`rtasidagi kommunikatsiya operatsiyalarining murakkabligi quyidagi ifoda bilan mos ravishda baholanishi mumkin (model A),

$$m_{nd}(m)\text{q } m_n + m * m_k + m_c, \qquad (2.1)$$

Bu ko`rinishdagi baho ma`lumotlar yuborish yo`li birga teng bo`lganida paketlarni yuborish metodlari uchun nisbatdan kelib chiqadi, ya`ni $l = 1$ bo`lganda. Bunday yondashuv ehtimoli borligini aytib, shu bilan birga ko`rib chiqilayotgan model doirasida $t_n$ tayyorgarlik vaqti doimiy deb olinishini (yuboriladigan ma`lumotlar hajmiga bog`liq emas), $t_c$ xizmat xabarlarini yuborish vaqti yuboriladigan paketlar soniga bog`liq emasligini va hokazolarni sezish mumkin [55]. Bu taxminlar haqiqatga to`liq mos kelmaydi va modeldan foydalanish natijasida olinadigan vaqtincha baholar zarur aniqlikka ega bo`lmasligi mumkin.

Keltirilgan barcha jihatlarni hisobga olib, vaqt baholarini qurish sxemasi aniqlanishi mumkin;

Yangi kengaytirilgan model doirasida ikki protsessor o`rtasidagi ma`lumotlar yuborilish murakkabligi quyidagi ifodalar bilan mos ravishda aniqlanadi (model B):

$$t_{nd} = \begin{cases} t_{b_0} + m \cdot t_{b_1} + (m + V_c) \cdot t_k, & n = 1 \\ t_{b_0} + (V_{max} - V_c) \cdot t_{b_1} + (m + V_c \cdot n) \cdot t_k, & n > 1 \end{cases}, \qquad (2.2)$$

bu yerda $n=\lceil m\ (V_{max}\text{-}V_c\ )\rceil$ yuboriladigan xabar bo`linadigan paketlar soni, $V_{max}$ kattalik tarmoqda etkazilishi mumkin bo`lgan paketning maksimal o`lchami (Fast Ethernet tarmog`ida MS Windows operatsion tizimi uchun V_{max}= 1500 bayt), V_c esa har bir yuboriladigan paketlarda xizmat ma`lumotlar hajmi (TCP_IP protokoli, Windows 2000 OT va Fast Ethernet tarmog`i uchun V_c = 78 bayt). Shuningdek, keltirilgan nisbatlarda m_{b0} konstanta latentlikning apparat tashkil etuvchisini xarakterlashi va foydalanilayotgan tarmoq qurilmasining parametrlariga bog`liq ekanligini, $m_{b1}$ qiymat tarmoq bo`yicha yuborilish uchun ma`lumotlarning bir baytini tayyorlash vaqtini berishini aytib o`tamiz. Buning natijasi sifatida, latentlik kattaligi:

$$t_n = t_{b_0} + v \cdot t_{b_1}, \qquad (2.3)$$

yuboriladigan ma`lumotlar hajmiga qarab chiziqli oshadi. Bunda ikkinchi va barcha navbatdagi paketlarni yuborish uchun ma`lumotlarni tayyorlash tarmoq bo`yicha oldingi paket va latentlik yuborilishi bilan birlashtirilishi mumkin deb taxmin qilinadi, shunda u quyidagi kattalikdan osha olmaydi:

$$t_n = t_{b_0} + (V_{max} - V_c) \cdot t_{b_1} \qquad (2.4)$$

Latentlikdan tashqari, taklif qilinayotgan ifodalarda kommunikatsion operatsiyani murakkabligini aniqlash uchun ma`lumotlarni yuborish vaqtini hisoblash qoidasi ham aniqlashtirilgan:

$$(m + V_c \cdot n) \cdot t_k \qquad (2.5)$$

Bu endi xizmat axboroti qo`shilishi hisobiga yuboriladigan paketlar soni o`sishida yuboriladigan ma`lumotlar hajmining oshish effektini hisobga olish imkonini beradi (paket sarlavhalari).Kommunikatsion operatsiyalar murakkabligini teoretik baholanishini qurish muammosining tahlilini yakunlagan holda, keltirilgan modellarni amaliy qo`llash uchun foydalanilayotgan nisbatlarning parametrlar qiymatining baholanishini bajarish zarur ekanligini aytib o`tish lozim [56]. Bu jihatdan ma`lumotlar yuborishga vaqt sarflarini hisoblashning ancha oddiy usullaridan foydalanish ham foydali bo`lishi mumkin – bunday ko`rinishdagi ma`lum sxemalar orasida, klasterning ikki protsessor tugunlar orasidagi

kommunikatsiya operatsiyalarining murakkabligi quyidagi ifoda bilan aniqlanadigan yondashuvni aytish mumkin (model C)

$$t_{pd}(m) = t_n + m/R, \qquad (2.6)$$

bu erda *p* ma`lumotlar yuborish tarmog`ining o`tkazish xususiyati.

Ma`lumotlar yuborishning real jarayonlariga ko`rib chiqilgan modellarning to`g`riligini tekshirish uchun Nijegorod universiteti ko`p protsessorli klasteri tarmog`ida o`tkazilgan tajribalar natijasini keltiramiz (IBM PC Pentium 4 1300 Mgts, 256 MB RAM, 10G`100 Fast Ethernet kompyuterlari). Tajribalarni o`tkazishda kommunikatsion operatsiyalarni amalga oshirish uchun quyidagi kutubxonadan foydalanilgan MPI.

Tajribalarning bir qismi modellar parametrlarini baholash uchun bajarildi:

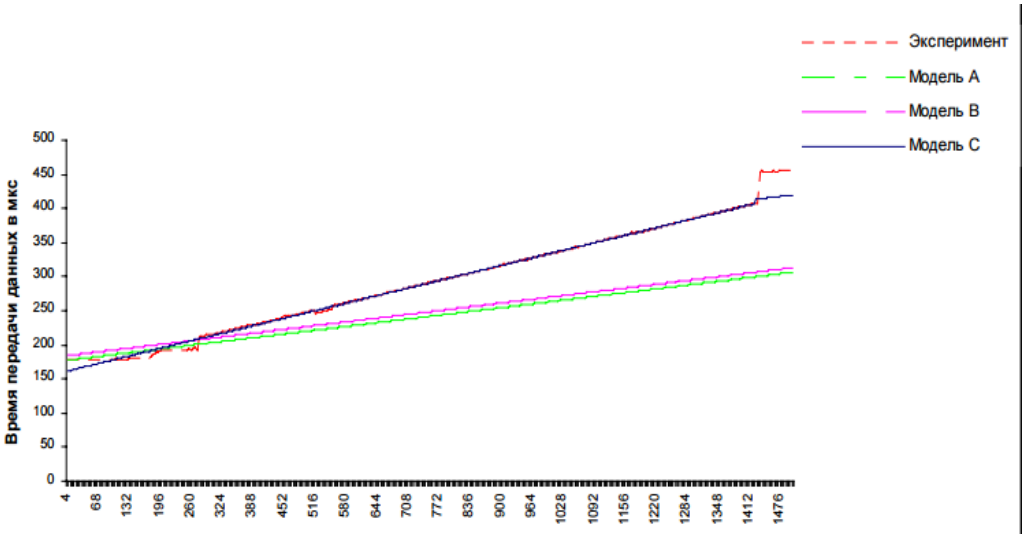
- A va C modellar uchun t_n latentlik qiymati nol uzunlikdagi xabarlarni yuborilish vaqti sifatida aniqlandi;

- R o`tkazish xususiyatining kattaligi tajribalarida kuzatiladigan maksimal ma`lumotlar yuborish tezligi bilan o`rnatildi, ya`ni

$$R = \max(t_{pd}(m)/m), \qquad (2.7)$$

- m_{b1} va m_{b1} kattaliklarning qiymati 0 dan V_{max} gacha o`lchamdagi xabarlarni yuborish vaqti chiziqli approksimatsiyasi yordamida baholandi.

Tajribalar davomida klasterning ikki protsessori o`rtasidagi ma`lumotlar yuborilishi amalga oshirildi, yuboriladigan xabarlarning o`lchami 0 dan 8 Mb gacha. Ancha aniq baholarni olish uchun har bir operatsiyaning bajarilishi ko`p marta amalga oshirildi (100000 martadan ko`p), shundan so`ng vaqtincha o`lchamlar vaqti o`rtalashtirildi. Ko`rsatish uchun quyida bir tajriba keltirilgan, u o`tkazilganida yuboriladigan xabarlar o`lchami 4 bayt qadam bilan 0 dan 1500 baytgacha o`zgargan.



2.13- rasm. Tajriba vaqti va ma`lumotlar hajmidan A, B, C modellar bo`yicha olingan vaqt bog`liqligi.

III.BOB. MEHNAT MUHOFAZASI VA TEXNIKA XAVFSIZLIGI

3.1. Kompyuterda ishlash vaqtida insonning charchash sabablari.

Ma’lumki kundalik hayotimizni shaxsiy kompyuterlarsiz tasavvur qilish qiyin. Shaxsiy kompyuterlar hozirgi kunda hayotimizning barcha tarmoqlariga kirib kelgan va muvaffaqiyatli qo’llanilmoqda. Ammo har bir qurilmadan foydalanishda xavfsizlik talablariga rioya qilish bu hayot talabi.

Shaxsiy kompyuterlardan foydalanishda ham xavfsizlik qoidalariga rioya qilish bu kishi organizmini turli xavfli omillardan zarar topishi yoki jarohatlanishini oldini oladi.

Eng avvalo kompyuterlardan foydalanishda ular uchun sanitariya va gigiena talablariga mos holda bino tanlash va ularni yong’in xavfsizligi vositalari, birinchi tibbiy yordam qutichalari bilan jihozlash maqsadga muvofiqdir.

O’zbekiston Respublikasi Mehnat va aholini ijtimoiy himoya hamda Sog’liqni saqlash vazirliklari tomonidan belgilangan talablar asosida, kompyuterlardan professional foydalanuvchilar ishga kirishdan oldin va davriy ravishda tibbiy ko’riklaridan o’tishlari shart. Homilador va emizikli farzandi bor ayollar uchun kompyuterda ishlash zararli.

Ish o’rinini to’g’ri tashkil qilish, kishilarni turli kasbiy kasallanishlarga yo’liqishini oldini olishda asosiy omil bo’lib hisoblanadi.

Shunday qilib, ishlab chiqarishda xavfsiz va zararsiz mehnat sharoitini yaratish, har bir rahbar va mas’ul xodimning vazifasi bo’lib, bunda nafaqat kishilarga sog’lom ish muhiti yaratiladi, balki ishlab chiqarishda samaradorlikni oshishiga ham olib keladi.

Zamonaviy avtomatlashtirilgan ishlab chiqarishda inson operatorning psixologiya va fiziologiyasi asosiy rolni egallaydi. Ishlab chiqarishda mehnat sharoitini yaxshilash va ilmiy asosda aniqlash, mehnatni to’g’ri ishlash maromini ta’minlash, mehnat tartibi va dam olishni tashkil qilish zarur.

Kompyuter bilan ishlash vaqtida inson quyidagi faktorlardan charchaydi:

- ekranni yorug’ligi;

- kontrast va fon o’rtasidagi aniqligi;

- kompyuterda ishlash paytidagi issiqlikdan nurlanishi;

- kompyuterda nurlanishning insonga ta’siri;

- kompyuter buzuqligi.

Displey bilan ishlaydigan EHM operatorlarida asosan, bosh og’rishi, bel, elka, orqa og’rishi, ko’z charchashi kuzatiladi.

Mehnat sharoitini yaxshilash maqsadida tashkiliy, gigienik, texnikaviy chora-tadbirlar ishlab chiqiladi va ishchi-xizmatchilar orasida mehnat gigienasi norma, qoidalariga rioya etish bo’yicha tashviqot ishlari olib boriladi.

Ishlab chiqarish sanitariyasi sanitariya-texnologik, tashkiliy tadbirlarni ifodalaydi va ishlab chiqarishda sog’lom mehnat sharoitlarini ta’minlaydi. Shu maqsadda ishchi-xizmatchilarning salomatligiga ta’sir qiluvchi texnologik jarayon va uskunalardagi kamchiliklarni yo’qotish yo’llarini ishlab chiqardi. Buning uchun sanoat korxonalarida texnika taraqqiyoti yutuqlaridan unumli foydalanishni, jarayonlarni olisdan boshqarish va ishchilarni zararli muhitda ishlashlarining oldini olishni, uskunalarni, qurilmalarni ochiq maydonda joylashtirishni, havo tarkibini tekshirib turishni, qo’l mehnatini talab qiladigan ishlarda imkoni boricha mexanizasiya vositalari va zamonaviy uskunalarni qo’llashni, himoya vositalaridan foydalanishni zarur deb hisoblaydi.

Ishlab chiqarishda xavfsizlikni ta’minlashda ergonomikaning ham ahamiyati katta. Ergonomikada insonning mehnat faoliyati jarayonida qulay, xavfsiz sharoitlarni yaratishga, mehnat unumdorligini oshirishga bog’liq bo’lgan imkoniyatlar o’rganiladi.

Bajarilayotgan turli jarayonlar va unga bog’liq bo’lgan uskuna, qurilmalar doirasida axborotni etkazuvchi-ko’rsatuvchi moslama-mashina modeli bo’lsa, operator murakkab tizimda bo’lsa ham, boshqarish ishlarini amalga oshiradi. Bu vazifani bajarish uchun shunday axborot modeli yaratilishi kerakki, bu model o’z vaqtida mashinaga taalluqli ta’rifni berishi, natijada operator toliqmasdan, fikrlab va e’tibor bilan axborotni xatosiz qabul qilib qayta ishlashi lozim. Murakkab hisoblangan vazifani yechish operatorning xavfsizligiga, aniq sifatli ishlashiga,

mehnat unumdorligiga, shuningdek insonning psixofiziologik imkoniyatlarini axborot modeliga mos bo'lishiga bog'liqdir.

Biofizik moslik operatorning ish qobiliyatini, normadagi fiziologik holatini ta'minlaydigan atrof-muhitning yaratilishini ifodalaydi. Insonnning kuchi va energetik qobiliyati ma'lum chegaraga ega. Shuning uchun ish jarayonida boshqarish tizimida charchash maqsadga muvofiq bo'lmagan oqibatga olib kelishi mumkin. Energetik moslik esa operatorning optimal imkoniyatlari asosida talab qilinadigan kuch, sarflanadigan quvvat, harakatning aniqligi va tezligi bilan mashinani boshqarilishidagi kelishuvni ifodalaydi.

Fazoviy-antropometrik moslik inson tanasi o'lchami, tashqi fazoning ta'sirli imkoniyatlari, ish jarayonida operatorning vaziyati, gavdaning turishi hisobga olinishini ifodalaydi. Vazifaning to'g'ri hal qilinishida ish joyi hajmi, operator harakatlanadigan masofa, balandlik, boshqaruv pultigacha bo'lgan oraliq va boshqa ko'rsatkichlar aniqlanadi.

Bulardan tashqari insonning psixik faoliyati ham muhim o'rin tutadi. Insonning qobiliyati, samarali mehnat faoliyati uning psixik kuchlanish darajasiga bog'liq. Operator uchun normal sharoitdagi his-tuyg'u va mehnat qilishi uchun ruhiy kuchlanish darajasi 40-60h dan oshmasligi ko'zda tutiladi, aks holda bu uning ish qobiliyatining pasayishiga olib keladi.

Mehnat sharoitini yaxshilash chora-tadbirlar orasida eng asosiysi ishchining ish holati va ish maromidir.

3.2. EHM operatori ish qobiliyatini saqlash va mehnat unumdorligini oshirish

Tananing ish qobiliyati birinchi galda markaziy asab tizimining holatiga bog'liq, markaziy asab tizimiga esa ijtimoiy muhit sharoitlari katta ta'sir ko'rsatadi.

Ish kuni va haftasining davomiyligini qisqartirish toliqishning oldini olishda eng muhim vosita hisoblanadi.

Ko’p mehnat talab qiladigan ishlarni mexanizatsiyalashtirish, yarim avtomat va avtomatlashgan texnologiya jarayonlariga o’tish mehnatni engillashtirib, ishlab chiqarish muhitini birmuncha qulay sharoitga keltirib, toliqish rivojlanishining oldini olishda katta ahamiyat kasb etadi.

Ishlab chiqarishda toliqishga qarshi kurash ko’pgina yo’nalishlar bo’yicha amalga oshirilib, ular orasida so’nggi yillarda ergonomika, ishlab chiqarish estetikasi kabi yangi yo’nalishlar ham vujudga keldi.

Ishlab chiqarish ta’limi jarayonida mashq qilishdan foydalanish ish qobiliyatini oshirishga kiradi. Muntazam mashq qilib borish unumli ishlashning eng ishonchli usuli hisoblanadi. Mashq jarayonida ishdagi xatti-harakatlar takomillashadi, ular birmuncha tartibli va tejamli bo’lib qoladi. Muntazam mashqlar tanada qator ijobiy siljishlar yuz berishiga olib keladi: mushak kuchi va chidamlilik oshadi, yurak-tomirlar va nafas tizimi faoliyati yaxshilanadi. Aqliy mehnatda mashqlar xotira, diqqat, iroda kabilarning takomillashuviga imkon beradi.

Mehnatni ilmiy asosda tashkil qilish, mehnat unumdorligini oshirishning asosiy vositasi hisoblanadi. Bunda avvalo, eng zamonaviy texnologiyadan, mashina, mexanizmlar va boshqa jihozlarning mukammal turlaridan foydalanishga, mehnatni to’g’ri tashkil qilishga asoslaniladi. Ayni vaqtda mehnat fiziologiyasi va ruhiyati talablarga rioya qilish uning ajralmas qismidir.

Asosiy fiziologik talablarga mehnat maromi, mehnat va dam olishning samarali tartibini tashkil etish kiradi. Bir maromdagi mehnat — smena, hafta, oy, yil maboynida bir tekisda bajariladigan mehnatdir. Mehnatning maromliligiga talab qo’zg’alish va tormozlanish jarayonlarini to’g’ri navbatlashda asab markazlarining fiziologik xususiyatlarini hisobga olish asoslangan. Maromli mehnat asab va mushak quvvatiyy oqilona sarflash, mehnat faoliyatining hamma davrlarida ish qobiliyatini quvvatlab turish imkonini beradi.

Uskunalarning nosozligi, materiallar, asbob-uskunalarning bo’lmasligi sababli ishdagi majburiy tanaffuslar ish qobiliyatiga salbiy ta’sir ko’rsatadi. Mehnat maromining buzilishi ishga berilish bosqichida erishilgan natijani

yo’qotishga sabab bo’lib, ish qobiliyatining boshlang’ich bosqichini birmuncha past darajaga qaytaradi. Ayni vaqtda ishda tez-tez bo’lar-bo’lmasga tanaffus qilaverish salbiy ehtiroslarni keltirib chiqaradi, bu ish qobiliyatini pasaytiribgina qolmay, balki ko’p takrorlanaverganda yurak-tomirlar patologiyasi rivojlanishiga sabab bo’lishi mumkin. Oylik va uch oylik rejalar nomuntazam bajariladigan (oy va chorak oxirida shoshilinch ishlar) korxonalarda ishchilar o’rtasidagi kasallanishni tahlil qilish ortiqcha charchash, shoshma-shosharlik va asab buzilishiga olib keladigan sharoit, ishdan keyin qolib ishlash soatlarining ko’payishi va hatto dam olish kunlarida ishlash surunkali kasalliklar, shamollash kasalliklarining o’sishiga olib kelishini ko’rsatdi. Kasallanish hodisalarining ko’p qismi «ishga hujum qilingan» oydan keyingi birinchi o’n kunlikka to’g’ri keladi. Qator korxonalarda maromli ishni joriy qilish mehnat unumdorligining 18—20%ga oshirishga va umumiy hamda kasbga doir kasallanishning pasayishiga olib keladi.

Mehnat va dam olishning oqilona tartibini belgilash, ish qobiliyatini yuksak darajada saqlab turishning eng muhim sharti hisoblanadi. Mehnat tartibi deganda ish va dam olish davrlarini taqsimlash tushuniladi. Smenaning muayin davrlariga, fiziologik jihatdan asoslangan ma’lum muddatli tanaffusni (tushki tanaffusdan tashqari) kiritish va ulardan oqilona foydalanish ish qobiliyatini yuqori darajada saqlab turish muhimdir. Bunday tanaffuslar toliqishning boshlang’ich bosqichiga to’g’ri kelsa va ishga berilish holatini buzmasa (uzoq davom etishi sababli) g’oyat foydali bo’ladi.

Qo’shimcha tanaffuslarni belgilash vaqti va ularning qancha muddat davom etishi ishning xususiyatiga bog’liq. CHunonchi, ish nechog’lik og’ir va jadal bo’lsa, smena boshlanganidan so’ng shuncha ertaroq (yoki kunning ikkinchi yarmi uchun — tushki tanaffusdan keyin) qisqa muddatli tanaffus, ayrim hollarda ikki yoki uch tanaffus joriy qilinadi. Ularning davomiyligi ham turlicha: 5-10 dan 15-30 minutgacha, bunda ish nechog’lik og’ir va jadal bo’lsa tanaffuslarning muddati shuncha davomli bo’ladi.

Tanaffuslar vaqtidagi dam olishni oqilona uyushtirilishida ishlab chiqarish gimnastikasini o’tkazish maqsadga muvofiq, bu toliqishni kamaytiradi va mehnat unumdorligini 3-15 %ga oshiradi. Bunday unumli dam olish loqayd dam olishga qaraganda birmuncha ta’sirchandir. CHunki faol dam olish davrida induktsiya yo’li bilan ishlayotgan markazlardan charchagan asab hujayralarining tormozlanishi chuqurlashadi, ularning birmuncha tez va to’liq tiklanishi ro’y beradi. Biroq og’ir mehnatda yoki havo harorati yuqori sharoitda ishlash hollarida shamollatiladigan xonada sust dam olish maqsadga muvofiq.

Toliqish profilaktikasida so’nggi vaqtda ergonomika (grekcha eggop — ish, pomos — qonun) degan nom bilan yangi yo’nalish vujudga keldi, bu fan mehnat unumdorligini oshirish, sog’liqni muhofaza qilish, ishda xavfsizlikni va qulay sharoit (kamfort)ni ta’minlash maqsadida odamni ishga moslashtirish uchun boshqa qator fanlarning ma’lumotlaridan foydalanishga asoslangan. Mashinalar va boshqa uskunalarni, jihozlarni ixtiro qilishda, ish joylarini uyushtirish va rejalashtirishda fiziologik va psixologik talablarga rioya qilish ergonomikaning asosiy yo’nalishlaridan biri hisoblanadi. Mashinalarni ixtiro qilishda ishlayotgan kishining ortiqcha harakatlardan holi etish, turli noqulayliklarga barham beradigan choralar ko’zda tutilishi kerak. CHunonchi, ozgina engashib ishlashda quvvat sarfi atigi 22% ga oshsa, ko’proq engashib bajaradigan ishda 45% ga oshadi. Boshqaruv qo’l va oyoq bilan amalga oshiriladigan hollarda odamning oyoq va qo’llari uchun mo’ljallangan ish maydonining samarali o’lchamlarini hisobga olish lozim.

Kuchni tejash tamoyiliga amal qilish ham muhimdir. Odam ko’p kuch sarflanadigan ishni uzoq vaqt mobaynida bajara olmaydi; ayni vaqtda muskul kuchi imkon bo’lganidan uzoq vaqt saqlanib turmasligi aniqlangan. Odam kuchining ta’sirini samarali yo’naltirishni nazarda tutish kerak. Tik turgan holatda eng yuqori kuchning o’ziga tomon qilingan harakatida rivojlanishi uzatilgan qo’lga nisbatan bukilgan qo’lda bosimning kuchi ko’proq ekanligi ma’lum. Asboblar, boshqa turli boshqaruv uskuna dastasini qulay hajm va shaklda yaratish yo’li bilan ham kuchni tejashga erishiladi. Toliqish profilaktikasida samarali ish vaziyati va to’g’ri qurilganligi katta ahamiyatga ega.

Muskullarning eng past darajadagi tarangligi hisobiga yuzaga keladigan erkin, odamdan kuch talab qilmaydigan holatga oqilona vaziyat deyiladi. Gavdaning tik yoki bir oz egilgan (ko'pi bilan 10-15 daraja) holatida shunday bo'ladi. Ishni o'tirgan, tik turgan holatda. ba'zan esa goh o'tirib, goh tik turib (o'tirib, turgan holda) bajarish mumkin. O'tirish vaziyatida statik harakatlar kamroq bo'lsa-da, ish vaqtidagi harakatlar ko'lami va sarflanadigan kuch unchalik katta bo'lmaganda (5 kg.gacha) uni qo'llash mumkin. Kuch 10 kg.ga etsa, o'tirib-tik turib ishlash holati, bundan katta bo'lganda esa tik turib ishlash holati maqsadga muvofiq.

O'tirib ishlash vaziyatida statik harakatlarni pasaytirish uchun ish jihozlari: stol, stul, oyoqlar uchun tirgakning fiziologik jihatdan asoslangan loyihalaridan foydalaniladi. Stulning baland-past bo'lishini boshqarib turish, suyanchig'i, tirsakni qo'yish uchun moslamasi borligi, o'tirib ishlash vaziyatdagi charchoqni kamaytiradi. Qator hollarda ishchi ro'parasidagi ish yuzasini yarim doira qilib qirqish, uni qiya qilib qo'yish maqsadga muvofiq. Fiziologik jihatda o'tirib-tik turib ishlash vaziyati g'oyat maqsadga muvofiq, u ishchiga o'zi uchun qulay vaziyat tanlash, qon dimlanib qoladigan sohalarda qon aylanishini tiklash imkonini beradi. Bir maromdagi ishlarni bajarishda bunday vaziyat ayniqsa foydali, chunki vaziyatni o'zgartirish ruhiy jihatdan xilma-xillikni vujudga keltiradi.

Ishlab chiqarish estetikasini joriy qilish: xonalarni ko'zni qamashtirmaydigan bo'yoqda bo'yash, yoritish, musiqa, intererni bezash toliqishning oldini olishda ruhiy fiziologik yo'nalish hisoblanadi. Ko'pchilik ishlab chiqarish binolarini yashil rangga bo'yash maqsadga muvofiq, chunki bu rang ta'sirsiz bo'lib, markaziy asab tizimini uyg'otishga ham, tormozlashga ham sabab bo'lmaydi. Asabga tormozlovchi ta'sir ko'rsatadigan ko'k va havo rang bo'yoqlar bilan issiqlikni ko'p ajratadigan yoki shovqin hosil qiladigan xonalarni hamda uskunalarni bo'yash maqsadga muvofiqdir. Qizil va sariq ranglar ko'zga ta'sir ko'rsatadi, shuning uchun ulardan ishchilar qisqa ishlay! digan sozlash ishlarini bajarish vaqtidagina bo'ladigan xona) larda foydalanish mumkin.

Biroq xona va uskunalarni bir xil rang bilan bo’yash yaramaydi, chunki bunday bir xillik odamga salbiy ta’sir etib, inson organizmi himoyasini zaiflashtiradi. Bo’yoqlardan, shuningdek, ishora-ehtiyotkorlik maqsadida ham foydalaniladi, transport vositalari, tsexdagi jo’mraklar va boshqa uskunalarni tiniq ranglarga bo’yash ishlab chiqarishda shikastlanish hollarining kamayishiga olib keladi. TSex va boshqa joylarni unumli yoritish yorug’likning bir tekis yoyilishi, tsexning ichki tomonini badiiy bezatish, chiroyli va qulay ish kiyimi toliqishning oldini oladi. Mehnat unumdorligini oshirishning muhim psixofiziologik vositasi jamoada do’stona munosabatlarni o’rnatish hisoblanib, bunda rahbarning o’rni muhim. Salbiy ehtiroslarni bartaraf etish toliqishning, balki asab va yurak-tomir kasalliklari paydo bo’lishining ham oldini oladi.

Mehnatning tibbiy sharoitlarini yaxshilash, ishlab chiqarish muhitining gigienik talablarga muvofiq kelishi mehnat unumdorligini oshirish yo’llaridan biri hisoblanadi. CHang, gaz, shovqin va tebranishni kamaytirish, me’yoriy mikroiqlim yaratish, bularning hammasi kasbga aloqador va kasbga aloqasi bo’lmagan qasalliklarning oldini olish uchungina emas, balki ish qobiliyatining yuksak bo’lishi uchun ham zarur shart hisoblanadi.

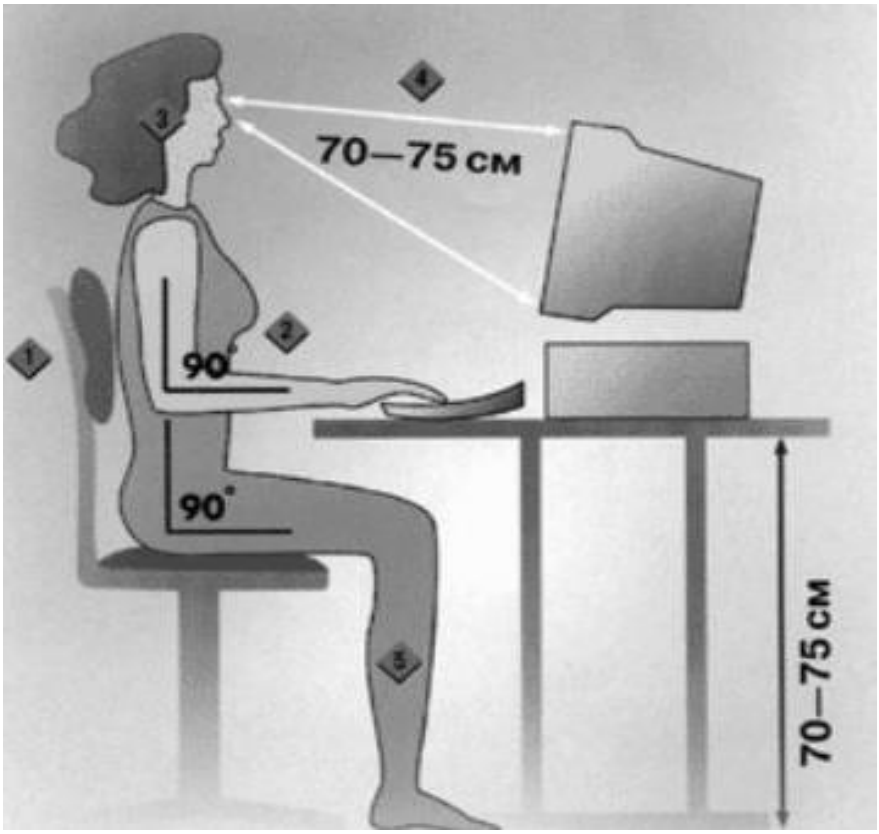
Aqliy mehnatda ish qobiliyatini yuksak darajada tutib turish uchun quyidagi qator sharoitlarga rioya qilish: uxlashdan yoki dam olishdan so’ng mehnat jarayoniga asta-sekin kirishish, mehnat faoliyatida dam olish davrlarini to’g’ri rejalashtirish. Bu sharoitlardan tashqari, ish joyining qulayligi, gavda vaziyatini vaqtida o’zgartirib turish imkoniyati borligi hamda ish sathining bir tekis yoritilishi ham muhim ahamiyatga ega.

Displeylar bilan ishlaydigan operatorni ish maromini yo’lga qo’yish mehnat xavfsizligini va charchashni kamaytirishni oldini oladi.

Ish kuni mobaynida ishchining ishlash qobiliyati birdan o’zining eng katta qiymatiga erisha olmaydi. Operatorning displey bilan ishlaganda ish maromi har 2 soatda yarim soat dam olishi yoki 1 soatda 15 minut dam olib turishi lozim. Bu qisqa-qisqa dam olish ish qobiliyatini yaxshilash va charchashni oldini oladi.

Bundan tashqari, operator va mashina o’rtasidagi masofa, shu masofaga binoan ekrandagi yozuvlarni kattaligi, ekran yorqinligi ham shular jumlasidandir.

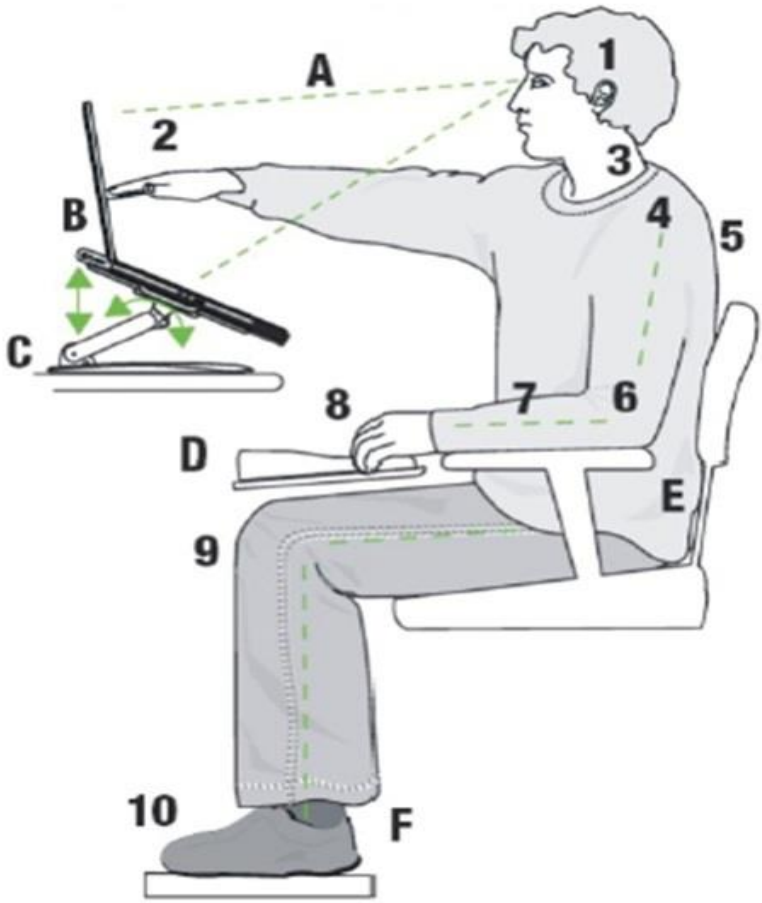
Ko’z va ekran oralig’i 60-80 sm, kattaligi esa 3-4 mm, optimal kenglik va balandlik 3:4, belgi orasidagi masofa esa uning bo’yidan 15-20 h bo’lishi kerak.



3.1-rasm. Operatorga kompyuterda ishlash paytidagi talab.

a b

3.2-rasm. Kompyuterda ishlaganda umurtqa pog’onasini to’g’ri (a) va noto’g’ri (b) holati.



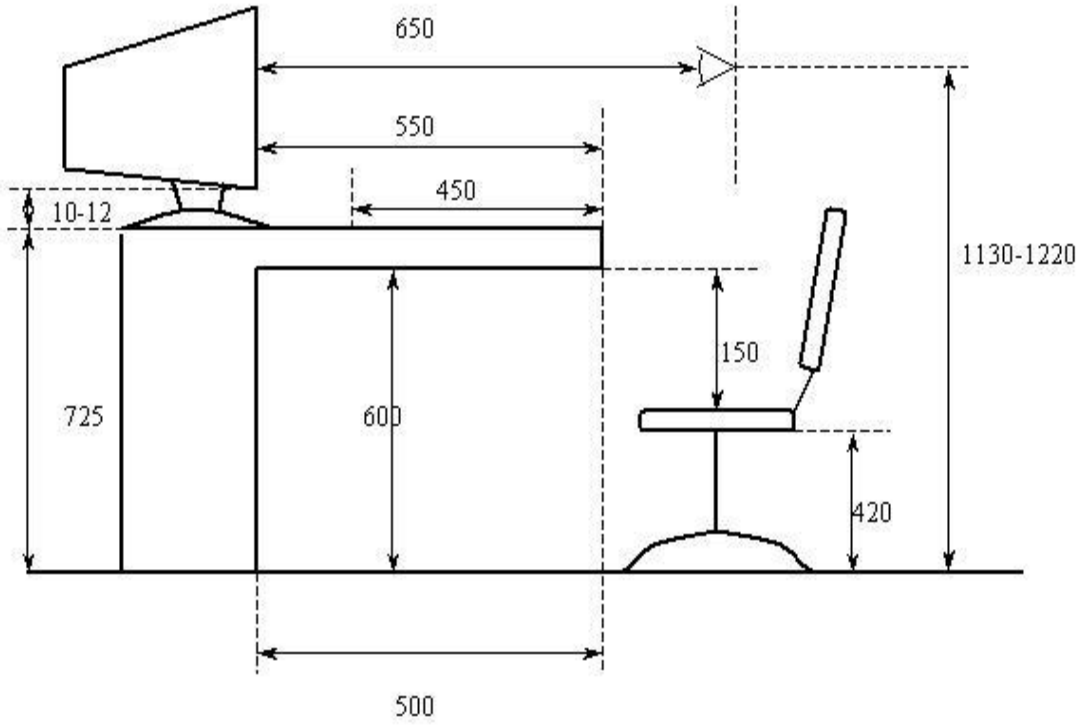
3.3-rasm. Ergonomik boshqaruv

3.3. Operatorning ish o’rniga qo’yiladigan talablar.

Ish holatini yaxshilashda operatorning ish o’rni asosiy ahamiyat kasb etadi. Ishchi stol qattiq holda bo’lishi kerak, chunki kerakli paytda ekranni, klaviaturani, dokumentlarni siljitish imkonini bersin. Ishchi stoli va o’tirgichlar bir-biriga mutanosib bo’lishi kerak.

O’tirgichni poldan balandligi 42-55 sm bo’ladi. Ishchi kursi ish davomiyligiga qarab tanlanadi: uzoq vaqt davomida ishlansa og’ir, kattakon kursi, qisqa vaqt davomida engil kursidan foydalansa bo’ladi, chunki ularni joyidan oson siljitiladi. Kursining tag qismi 5 ta tayanchdan iborat bo’lishi zarur.

O’tirgichlar qulay bo’lishi kerak, uning o’lchami 40x40 sm.kv. dan oshmasligi zarur.



3.4-rasm. Operator ish o’rniga qo’yilgan talab.

Ish joyini rejalashda stolda joylashgan displey va klaviatura turishiga ko’ra qo’l holatini ham nazarda tutish kerak.

Operator ish o’rnida displey joylashtirilayotganda ko’z va boshning harakati aniqlanadigan ko’rish maydoni ham ko’zda tutiladi.

Operator o’z ish joyini shunday ta’minlashi kerakki, bunda ekran o’rtada, yozilayotgan dokument esa chap yonda yoki maxsus joylatirgichlarda turishi kerak. Klaviatura nisbatan tekis turadi, o’rta qatordagi klavishalar balandligi 2.5-5.0 sm . Klaviaturaning o’rta qismidan stol qirg’og’igacha bo’lgan masofa 16 sm.

EHMlar ish davomida o’zidan turli rentgen nurlarini chiqaradi. Shuni hisobga olib, ishlash davomiyligini qisqartirish kerak, shuningdek, maxsus himoya ekranlaridan foydalanish kerak.

Ishlab chiqarish binolarini normadagi metereologik va sanitariya-gigiena sharoitlari bilan ta’minlashda, ish jarayonida zararli va zaharli mahsulot-moddalarning miqdorini chegaralangan darajada bo’lishida, mehnat sharoitlarini yanada sog’lomlashtirishda, mehnat unumdorligini va mehnat xavfsizligini oshirishda shamollatish katta ahamiyatga ega.

Binodagi havo almashinishi ma’lum miqdorda bo’lishi uchun devor, deraza, yopma va fonarlardagi darchalar ko’proq yoki kamroq ochilib, shamol yo’nalishiga qarab moslamalar yordamida boshqariladi.

Qish vaqtida binodagi va tashqi havo haroratidagi farq katta bo’lganligi uchun, binodagi havoni almashtirish kamroq talab qilinadi. Shuning uchun ham havo beruvchi darchalarning yuzasi kamaytirilib, ular pol yuzasidan 5-6 metr balandlikda o’rnatiladi.

Yoz faslida esa havo oqimi 1,5-2 metr balandlikda uyushtirilsa etarli. Binolarda, xonalarda shamollatish qurilmalari ish boshlanishidan 10-15 minut avval ishga tushirilib, ish tamom bo’lganidan 10-15 minut keyin to’xtatiladi.

Sanitariya-gigiena talablariga mos keluvchi yana bir holat xonaning yoritilganlik darajasidir. To’g’ri va rejali yoritilgan xonalarda ish unumdorligi oshadi, toliqish kamayadi va korxonaning xavfsizligi ta’minlanadi. Yaxshi yoritilmagan xonalarda ishlayotgan operator yoki ishchi atrofda joylashtirilgan narsa va buyumlarni yaxshi ko’rmaydi, ishlab chiqarish sharoitiga moslasha olmaydi. Natijada ishchi mehnat faoliyatida ko’zning zo’riqishi vujudga keladi. Haddan tashqari yoritilganlik ham ko’zga yomon ta’sir ko’rsatadi.

Kompyuterlarni to’g’ri joylashtirish.

Kompyuterlarni belgilangan masofada joylashtirish, ularda hosil bo’ladigan elektrostatik va elektromagnit maydonlarini foydalanuvchilarga ta’siri xavfini kamaytiradi.

Binolarda kompyuterlarni joylashtirishda gigiena talablarga asosan har bir foydalanuvchi shaxs uchun eng qulay ish hududini ta’minlanish talabi hisobga olinishi shart, ya’ni bir kishi uchun ish joyining hajmi elektron-nurlanishli monitorli kompyuterlar uchun 20m “(ish joyi maydoni 6m2 dan kam bo’lmagan holda)” monitorli kompyuterlar uchun (ish joyi maydoni 4,5m holda) 15-20m3 kam bo’lmasligi kerak.

3.5-rasm. Ofisda va uyda kompyuterni joylashtirish.

Kompyuter xonasida stol va kursilarga talablar mavjud bo'lib, stol balandligi erdan 68-77 sm bo'lib, kursilar esa aylanuvchan bo'lishi kerak va albatta orqasida suyanchig'i bo'lishi kerak. Chunki stol-kursilar o'z gabariti bilan to'g'ri kelmasa, foydalanuvchi tezda charchab qoladi va zerikishga olib keladi. Stol va kursilar shunday joylashtirilishi kerakki, ular insonlarga turib yurishga xalaqit bermasligi kerak.

Ish stolining kengligi, oyoq ostidagi zinachaning me'yorida bo'lishi hamda aylanma va vertikal holati o'zgaradigan o'tirgich ishda ancha qulayliklar yaratadi.

3.6-rasm. Kompyuter uchun maxsus stol.

3.7-rasm. Burchakka qo’yiladigan stol.

Xonaning yoritilganligiga qo’yiladigan talablar.

Xonaning normal yoritilganligi 400 luks bo’lishi kerak. Yorug’lik ishchining ish joyiga qaysi tarafdan tushayotganligi ham muhim. Yoritish tizimi turlarini tanlash asosan bajarilayotgan ishning texnologik jarayoniga, kategoriyasiga bog’liq.

Kompyuterlarni shunday joylashtirish kerakki, bunda kompyuter monitori yuzasida tabiiy yoki sun’iy yorug’lik aks ta’siri bo’lmasligi lozim, aks holda bu holat kishi ko’zini tez toliqishiga va ko’rish qobiliyatini pasayishiga olib keladi.

Sun’iy yoritgichlar kompyuter monitorining yuza qismiga nisbatan chap (maxsus stol usti yoritgichlar) yoki orqa qismida (umumiy yoritish qurilmalari) o’rnatilishi va kompyuter monitorining balandligi ko’zning gorizontal ko’rish qismidan balandda o’rnatilgan bo’lishi kerak.

Kompyuter stolida o’rnatilgan yoritgichning yorug’lik miqdori 300-500 Lk (luks) va kompyuter monitorining ekranini yorug’lik miqdori 300 Lk dan va yoritilganlik quvvati 35kd (kandela) dan yuqori bo’lmasligi lozim. Ish joylardagi

yorug’likning miqdorini amalda qo’llaniladigan YU-116 luksometrlar orqali o’lchash mumkin.

3.8-rasm. Kompyuter uchun maxsus stol lampalari.

Kompyuter xonalarida mikroiqlim sharoiti.

Kompyuter monitorini va prosessorini havo almashuvchanlik tizimiga havoning erkin almashishi, issiqlik manbasidan uzoq masofada tutish, kompyuter ish samardorligini va ishonchliligini oshiradi. Tizim va elektr tormog’i o’tkazgichlarini o’zaro ulashda ularni o’ralib qolishiga yo’l qo’yilmaslik lozim.

Ishlab chiqarishda samardorlikni oshirishning asosiy omillaridan biri, bu xonalarda mikroiqlim sharoitni qulayligi, ya’ni ish joyidagi havoning harorati, nisbiy namligi va shamolning harakat tezligining GOST 12.001.005-86 talablariga mos kelishi. GOST 12.001.005-86 ga asosan kompyuterlar o’rnatilgan xonada havoning harorati 21...25°C, nisbiy namlik miqdori 40-60h va shamolning harakat tezligi 0,1m/s oshmasligi yoki tushmasligi lozim.

XULOSA

Ushbu bitiruv malakaviy ishida ma`lumotlarni tahlil qilish, saqlash va saralash amallarini optimallashtirish, xotira resurlaridan foydalanishda siqish algoritmlarining optimal usullarini yaratish va tasvirlarni qayta ishlashda ko`p yadroli protsessorlarning unumdorlik darajasini oshirishga mo`ljallangan parallel algoritmlarini yaratish kabi bir qancha amallar ketma-ketligi keltirilgan. Ishni bajarish davomida quyidagi natijalarga erishildi:

Ma`lumotlarni tahlil qilishda muammoli masalalarni tahlil qilgan holda siqish algoritmlarining optimal usullaridan foydalanish yaxshi natija berishi aniqlandi. Bunda tasvirlarni raqamli qayta ishlaganda veyvlet-jarayoni va Gauss usulidan foydalanish, qayta ishlash amallarida optimal yondashish mumkinligi aniqlandi. Buning natijasida xotira resurslaridan foydalanishda kamroq joy egallashini va axborotni xotiradan o`qish va qayta ishlab qayta xotiraga yozish jarayonlari tezlashtirishga olib keldi.

Ma`lumotlarni qayta ishlashda ko`p yadroli protsessorlarga mo`ljallangan parallelashtirish algoritmlarini qo`llash yaxshi samara berdi. C++ dasturlash tilidan foydalanildi va parallel algoritmni ta`minlab OpenMP va OpenCV kompilyatorlari direktivalaridan foydalanildi va ular yordamida protsessor unumdorlik darajasi oshdi.

Ish davomida tasvirlarni qayta ishlashda tasvir qiymatlarini baytli massivga o`zlashtirish, veyvlet-jarayonlarni amalga oshirganda oqimlarga ajratish usullari va xotirani parallel holda dinamik joy ajratish kabi jarayonlar bajarildi va yaxshi samaradorlik ko`rsatdi.

Tadqiqot natijalaridan kelib chiqqan ma`lumotlar yordamida parallelashtirishning umumiy usuli yaratildi. Tasvirlar zarrachalar sonini oshirish natijasida parallel algoritmlar effektivligini khrish mumkin bo`ldi, ba`zida esa butun tasvirni kvadrat qismlarga ajratgan holda alohida funktsiyalarga ajratish va protsessor oqimlar soniga teng amallarga bo`lib berish ham o`z unumdorligini ko`rsatdi.

Tasvir ma`lumotlarini parallel algoritim yordamida oqimlarga ajratish bilan qayta ishlash natijasida ko`p yadroli protsessor unumdorligini quyidagi xususiyat – tasvirning qayta ishlanilayotgan qiymatlar soniga bog`liq ekanligi natijalardan kelib chiqqan holda aniqlandi. Protsessor unumdorligi etarli darajaga etgunicha o`sib boradi (tasvirning 256 dan 4096 gacha). Optimal echim sifatida unumdorlik darajasi protsessorlarning yadrolar soni qiymatiga yaqinlashib boradi.

Parallel qayta ishlashda oqimlar sonining optimal soni protsessorning xisoblash darajasiga teng bo`lishi lozim. Ikta fizik yadroga ega protsessorlar 4 ta mantiqiy oqim yaratib bera olishi va unumdorlik darajasini haqiqiy 4 yadroli protsessor natijasiga yaqinlashtirib berishi mumkin.

Ushbu ishda yadrolar soni 2 va 4 ga teng bo`lgan turli protsessorlarda o`tqazilgan tajriba natijalari keltirildi. Natijalarga asosida quyidagi xulosalar chiqarildi.

Bitruv malakaviy ishining natijasi – ketma-ket va parallel algoritmlarni tasvirlarni veyvlet-jarayon hamda Gauss usuli yordamida siqishda ketgan vaqtini va unumdorlik darajasini aniqlaydigan dasturiy vosita yaratildi. Dastur 2 va 4 yadroli Intel Corei7 i i5 protsessorlarida Windows XP i Windows 7 va undan yuqori operatsion tizimlarda tajribalar o`tkazildi. Quyidagi dastur yordamida ko`pgina tajribalar o`tqazildi va tajriba natijalar asosida yakuniy xulosalar chiqarildi.

ADABIYOTLAR RO`YHATI

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб: «БХВ-Петербург», 2012. – 608 б.

2. Богачев К.Ю. Основы параллельного программирования. – М.: «БИНОМ. Лаборатория знаний», 2013. – 342 б.

3. Малышкин В.Э.. Основы параллельных вычислений: Учебное пособие. Часть 2. – Новосибирск: ЦИТ СГГА, 2012. – 264 с.

4. Шпаковский Г.И. Реализация параллельных вычислений: MPI, OpenMP, кластеры, грид, многоядерные процессоры, графические процессоры, квантовые компьютеры. – Минск: Белорусский Государственный Университет, 2010. – 155с.

5. Ярославский Л.П. «Введение в цифровую обработку изображений», Москва сов.радио, 2012й.

6. Грузман И.С. «Цифровая обработка изображений в информационных системах», Новосибирск 2012г.

7. Антонов А.С. Параллельное программирование с использованием технологии OpenMP. – М: издательство Московского Университета, 2011 г. – 77с.

8. Левин М.П. Параллельное программирование с использованием OpenMP: учебное пособие. – М: “Бином. Лаборатория знаний”, 2012г. – 118 с.

9. Старченко А.В., Есаулов А.О. Параллельные вычисления на многопроцессорных вычислительных системах. – Томск: Изд-во ТГУ, 2010. – 56 б.

10. Мелехин В. Павловский Е. Вычислительные машины, системы и сети. – Масква, Изд-во: «Академия», 2012. – 103б.

11. Шпаковский Г.И. Реализация параллельных вычислений: MPI, OpenMP, кластеры, грид, многоядерные процессоры, графические

процессоры, квантовые компьютеры. – Минск: Белорусский Государственный Университет, 2012. – 155б.

12. Левин М.П.: «Параллельное программирование с OpenMP» / Левин М.П., режим доступа: http://www.intuit.ru/department/se/openmp/

13. A. A. Kondratyev. The parallel image processing and clustering based on Kohonen maps by using clusters and graphics processing units // Proceedings of Junior research and development conference of Ailamazyan Pereslavl university. Pereslavl, 2012. p. . (in Russian).

14. Тищенко И. П. Параллельная кластеризация цветных изоб-Ражений на основе нейронной сети кохонена с исполь-зованием суперкомпьютера семейства .скиф., № 9: Нейро-компьютеры: разработка, 2013, с. 30–35.

16. Breiman L. Random Forests. // Machine Learning. 2001. V. 45, №. 1, P. 5- 32.

17. Breiman L., Friedman J., Olshen R., Stone C. Classification and Regression Trees. Wadsworth,2011.

18. Breiman L. Bagging predictors // Machine Learning. 2012. V. 26, № 2, P. 123-140.

19. Druzhkov P. N., Eruhimov V. L., Kozinov E. A., Kustikova V. D., Meyerov I. B., Polovinkin A. N., Zolotykh N. Yu. On some new object detection Features in OpenCV Library // Pattern Recognition and Image Analysis. 2011.V. 21, № 3. P. 384–386.

20. Enzweiler M., Gavrila D. M . Monocular Pedestrian Detection: Survey and Experiments // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2010. V.31, № 12. P. 2179–2195.