ҚАЗАҚСТАН РЕСПУБЛИКАСЫ
БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РЕСПУБЛИКИ КАЗАХСТАН
MINISTRY OF EDUCATION AND SCIENCE
OF THE REPUBLIC OF KAZAKHSTAN

Л.Н. ГУМИЛЕВ АТЫНДАҒЫ
ЕУРАЗИЯ ҰЛТТЫҚ УНИВЕРСИТЕТІ
ЕВРАЗИЙСКИЙ НАЦИОНАЛЬНЫЙ
УНИВЕРСИТЕТ ИМ. Л.Н. ГУМИЛЕВА
GUMILYOV EURASIAN
NATIONAL UNIVERSITY

ЖАС ҒАЛЫМДАР КЕҢЕСІ

Студенттер мен жас ғалымдардың «Ғылым және білім - 2015» атты X Халықаралық ғылыми конференциясының БАЯНДАМАЛАР ЖИНАҒЫ

СБОРНИК МАТЕРИАЛОВ X Международной научной конференции студентов и молодых ученых «Наука и образование - 2015»

PROCEEDINGS of the X International Scientific Conference for students and young scholars «Science and education - 2015»

УДК 001:37.0
ББК72+74.04
Ғ 96

Ғ96
«Ғылым және білім – 2015» атты студенттер мен жас ғалымдардың X Халық. ғыл. конф. = X Межд. науч. конф. студентов и молодых ученых «Наука и образование - 2015» = The X International Scientific Conference for students and young scholars «Science and education - 2015». – Астана: http://www.enu.kz/ru/nauka/nauka-i-obrazovanie-2015/, 2015. – 7419 стр. қазақша, орысша, ағылшынша.

ISBN 978-9965-31-695-1

Жинаққа студенттердің, магистранттардың, докоранттардың және жас ғалымдардың жаратылыстану-техникалық және гуманитарлық ғылымдардың өзекті мәселелері бойынша баяндамалары енгізілген.

The proceedings are the papers of students, undergraduates, doctoral students and young researchers on topical issues of natural and technical sciences and humanities.

В сборник вошли доклады студентов, магистрантов, докторантов и молодых ученых по актуальным вопросам естественно-технических и гуманитарных наук.

УДК 001:37.0
ББК 72+74.04

ISBN 978-9965-31-695-1

ӘОЖ 004.421.2

ПАРАЛЛЕЛЬ ПРОГРАММАЛАУ ЭЛЕМЕНТТЕРІН ОҚУ ҮРДІСІНДЕ ҚОЛДАНУ

Кайдарова Аяулым Базаргазыевна,
kaidarova_ab@enu.kz,
Мусапарбекова Молдир Сериккалиевна
musaparbekova_ms@enu.kz
Л.Н.Гумилев атындағы ЕҰУ
Ақпараттық технологиялар факультеті
Информатика кафедрасының оқытушылары, Астана, Қазақстан

Параллель алгоритмдерді өңдеу ойлау қызметінің әдістерін қалыптастырады. Осыған сәйкес информатика мамандығының студенттерін параллель программалауға оқытудың әдістемелік жүйесі мақсатты түрде студенттерде алгоритмдік ойлау қызметінің әдісін қалыптастыру қажет.

Жоғары мектептің оқыту процесінде соңғы озық технологиялардың бірі желіде программалауды пайдалану, соның ішінде параллель программалау элементтерін енгізу. Қазіргі уақытта есептеуіш техника мен программалық қамтамасыздандыру саласында екпінді өрлеу байқалады. Көппроцессорлы есептеуіш жүйелер ақпараттық қоғамның ғылыми және өндірістік қызметіне қарқынды түрде ене бастады. Параллель есептеулер мен параллель программалау есептеуіш математика мен программалау саласына жатады. Параллель программалау бір жағынан программаның құрылу әдісін ғана өзгертпейді, сонымен бірге адамның ойлау қызметін, параллельді ойлаудың ерекше стилін қалыптастырады. Сол себепті қоғамның талабына сай адам қабілеттілігін қазіргі заманғы көппроцессорлы жүйелер көмегімен үлкен көлемдегі ақпаратты өңдеу арқылы дамыту мектеп қабырғасынан бастап оқушыларға мақсатты түрде ойлау шеберлігі мен қызметін қалыптастыруды анықтайды. Осыған байланысты параллель есептеулер мен параллель программалаудың негізін оқыту болашақ информатика мұғалімінің кәсіптік дайындығының бөлігі болып табылуы керек. Бірақ қазіргі кезде осы тәріздес дайындық болашақ информатика мұғалімі мамандығында педагогикалық білім беру жүйесінде жүзеге аспаған, сондықтан осы бағытта тереңінен оқыту маңызды болып табылады [1].

Microsoft Parallel Extensions. Parallel Extensions to the .NET Framework (басқа атаулары - Parallel FX Library, PFX) – басқарушы код (managed) негізіндегі программаларды қолдану үшін жасалған Microsoft фирмасының кітапханасы. Ол арнайы координацияланатын (coordinating) берілгендер құрылымы қолданылатын есептерді параллельдеуге мүмкіндік береді. Сол арқылы PFX кітапханасы параллель есептеулердің жазылуын жеңілдетеді, қазіргі заманғы параллель программалау моделінің кейбір қиыншылықтарын алып тастау арқылы ядролар саны мен процессорлар саны өскен сайын өнімділігін арттырады. Кітапхананың алғашқы нұсқасы 2007ж 29 қараша шыққан болатын, келесі нұсқалары 2007ж желтоқсаны мен 2008ж маусымында жаңарған болатын.

Parallel Extensions параллелизмді ұйымдастырудың бірнеше жаңа тәсілдерін қамтамасыздандырады:

- Мәліметтерді декларативті өңдеудің параллелизмі. Параллель интегралданған сұраныс тілі (PLINQ) арқылы іске асырылады – LINQ параллель іске асырылуы. LINQ-тан айырмашылығы қолжетімді ядролар мен процессорлардың жүктелуімен бірге масштабталуын қамтамасыз ете отыра сұраныстар параллель түрде орындалады.
- Мәліметтерді императивті өңдеудің параллелизмі. For және foreach

сияқты циклдарының негізгі итеративті операциялардың параллель нұсқаларын кітапханалық іске асырылу көмегімен жүзеге асады. Олардың орындалуы автоматты түрде есептеуіш жүйенің барлық қолжетімді ядролар мен процессорларға бөлінеді.

Есеп деңгейіндегі параллелизм. Parallel Extensions кітапханасы аз салмақты есептер көмегімен параллель орындалатын кодты құрылымдай отыра жұмыс ағынын жоғары деңгейде орындалуын қамтамасыздандырады. Parallel Extensions кітапханасының

жобалаушысы осы есептердің орындалуын басқару мен диспетчерлендірілуін орындайды, сондай-ақ ерекше жағдайларды өңдеудің біркелкі механизмі. Parallel Extensions – бұл басқарылушы (managed) кітапхана және өзінің жұмысына .NET Framework 3.5 орнатылуын талап етеді [2].

Parallel Extensions-ты ең жақсы қолдану тәсілі. Parallel Extensions кітапханасы көпядролы процессор мен көппроцессорлы машиналарды қолданудың жоғары өнімділігін қамтамасыздандыру мақсаты үшін өнделді. Өңдеу барысында жұмысқа көп мөлшердегі ядроларды қолжетімділігіне қарай динамикалық түрде тарту арқылы параллель есептеулердің көлемі жоғары шекарасы жайлы шешімді қабылдаудың орнына кітапхана орындалатын есептерді автоматты түрде ауқымдайды. Көпядролы процессор мен көппроцессорлы машиналарға Parallel Extensions-ды қолданған кезде өнімділіктің өсімі өседі. Сонымен қатар Parallel Extension шығынды минималдау үшін және бірпроцессорлы машиналарда орындалу үшін жасалған.

Parallel Extensions негізгі, базалық тұжырымдамасы – бұл тапсырма(Task) – басқа тапсырмалардан тәуелсіз орындала алатын лямбда функциясы арқылы көрсетілген кодтың шағын бөлігі. PLINQ де, TPL API есептерді құру үшін әдістерді көрсетеді – PLINQ сұранысты үлкен емес бірнеше есептерге бөледі, ал TPL API - Paralel.For, Parallel.ForEach және Parallel.Invoke әдістері цикл немесе кодтың реттілік блоктарын есептерге бөледі.

Parallel Extensions есептердің орындалуын жобалайтын есептер менеджерінен тұрады. Есептер менеджерінің басқа атауы – жобалаушы. Жобалаушы көптеген жұмыс ағындарын басқарады, осы жұмыс ағындарында есептердің орындалуы жүреді. Үнсіздік бойынша процессорлар санына тең ағын саны құрылады. Әр ағын өз есебінің кезегімен байланысты. Кезектегі есеп орындалуы аяқталғаннан кейін ағын өзінің локальды кезегінен келесі есепті шығарып тастайды. Егер ол бос болса, онда есепті толтыру үшін локальды кезектегі келесі жұмыс ағынын ала алады. Есептер бір бірінен тәуелсіз орындалады. Сондықтан бөлінетін ресурстарды қолданғанда блокировка немесе басқа конструкциялар көмегімен синхронизацияны орындау қажет.

TPL (Task Parallel Library). TPL-ды өз қосымшаңызда қолданар алдында сіздің жобаңыз Sytem. Threading.dll. сілтемесінен тұратынына көз жеткізіңіз. TPL мүмкіншіліктері бұл кітапхана аттарының екі кеңістігіне негізделген: System.Threading и System.Threading.Tasks. Онда үш қарапайым типі анықталған:

- System.Threading.Parallel
- System.Threading.Tasks.Task
- System.Threading.Tasks.Future<T>

System.Threading.Parallel. System.Threading.Parallel класы .Net. қосымшаларда циклдар мен код блоктарының реттілігін параллельдеуге мүмкіндік береді. Бұл функционалдылық статикалық әдістердің жиыны ретінде іске асырылды, нақтырақ айтқанда For, ForEach және Invoke әдістері [11].

Parallel.For и Parallel.ForEach. Parallel.For және Parallel.ForEach конструкциялары for және foreach циклдарының параллель аналогы болып табылады. Олардың қолданылуы цикл итерациясының тәуелсіз орындалу жағдайында, яғни егер ешбір итерацияда алдыңғы итерациялардағы жұмыс нәтижесі қолданылмаса ғана дұрыс болады. Онда бұл итерациялар параллель орындала алады.

Parallel.Invoke. Invoke статикалық әдісі параллель жүзеге асыруда блок операторларының орындалуын параллельдеуге мүмкіндік береді. Көп жағдайда қосымшалар ішінде олардың орындалу реті маңызды емес операторлар реттілігі бар. Бұл жағдайда бірінен соң бірі орындалу орнына олардың параллель орындалу мүмкіндігі бар, бұл өнімділікті арттырады.

Parallel.For конструкциясы. Кей жағдайларда қолданылатын программалардың циклдарындағы итерациялар тәуелсіз болып табылады, яғни ешбір итерацияда алдыңғы итерацияның жұмысы қолданылмайды. System.Threading.Parallel класы арқылы мұндай циклдар қолжетімді ядролар (процессорлар) көлемі жеткілікті болған жағдайда әр итерация

параллель орындала алатын циклға айналуы мүмкін. Циклдардың параллель орындалуы тәуелсіз итерациялардың дұрыс емес нәтижеге әкелуі мүмкін. Бұл жағдайда цикл құрылымын қайта қарастыру қажет немесе синхронизация примитивтерін және мәліметтер құрылымын қолдану қажет.

PFX-те For әдісінің параллель орындалатын бірнеше нұсқалары бар. Ең жиі қолданылатыны For(Int32, Int32, Action<Int32>) болып табылады. Мұнда бірінші параметр бастапқы индексті береді, екінші параметр – соңғы индекс, соңғы параметр – әр итерацияда орындалатын іс–әрекетті анықтайтын делегат [12].

Үрдістердің орындалуын жобалау. Үрдістердің орындалуын жобалау көппроцессорлы есептеуіш жүйеге жататын теоретикалық және практикалық ұғымдардың бірі болып табылады. Үрдістерді немесе есептердің орындалуының негізі жобалау көптеген есептің және жүйедегі көптеген есептеуіш элемменттердің(ядро, процессор) жұмыс динамикасын ескергендегі құрылуы болып табылады. Осы немесе басқа есептің орындалу жобасының алгоритмін іске асыратын программаны жобалаушы(scheduler) деп атайды.

Жобалаушы әдетте процессорлардың максимум жүктелуін, жүйенің жоғары өтімділік қабілеттілігін қамтамасыз етеді – аз уақытта орындалған есептер көлемін, орындалу кезегіндегі есептің минималды уақыт табуын, есептер арасындағы процессорлы уақытқа бөлудің әділеттілігі.

Бірпроцессорлы есептеуіш жүйе ішінде әдетте келесі классикалық жобалау тәртібі қолданылады:

- FIFO – есептердің кезегі (First In, First Out)
- LIFO – есептердің кезегі (стек) (Last In, First Out)

LIFO және FIFO нұсқалары процессорге монопольды рұқсаты бар операциялық жүйе астында бірпроцессорлы жүйедегі үрдістердің орындалуының статикалық жобалауын иллюстрациялайды. FIFO кезек негізіндегі айналма қызмет ету [3].

Бұл нұсқа үрдісті ығыстыруы бар бірпроцессорлы жүйедегі жобалаушының жұмыс істеу үрдісін иллюстрациялайды, көп жағдайда белгілі басым үрдістерге байланысты үрдістердің сұрыпталуы қолданылады. Үрдіс басымдылығын есептеу немесе тағайындау принциптері жобалаушы типімен анықталады.

Болашақ информатика мұғалімдерді желіде программалау бойынша даярлау мәселесінің көкейкестілігі ғылыми-техникалық жетістіктердің қоғам өміріне жан-жақты ендірілуімен, жастарға желіде программалауды жетік меңгеруге байланысты ғылыми тұрғыда бірыңғай жүйе жасау қажеттілігімен және негізделген оның педагогикалық жасаумен айқындалады.

Болашақ информатика мұғалімдерінің желіде программалауды кәсіби іс-әрекеттерде пайдалану даярлығын қалыптастырудың тиімділігі педагогикалық шарттардың жиынтығы арқылы анықталады: болашақ мұғалімдердің тұлғалық субъективтік көзқарасын кәсіби іс-әрекеттерде оның желіде программалауды пайдалануға даярлығын қалыптастыру үрдісінде өзектендіру; болашақ мұғалімдердің кәсіби іс-әрекеттерде желіде программалауды пайдалануға даярлығын қалыптастыру үрдісіндегі басқаруды және өзін-өзі басқаруды жыддамдату; кәсіби іс-әрекеттерде ақпараттық-компьютерлік технологияны пайдалану теориясы мен практикасындағы оқу материалын құрылымдауды модульдік технологиялау.

Қолданылған әдебиет

1. Е.Е. Дүйсембиев, Параллель есептеулер. Жоғары өнімді технологиялар /. Алматы, 2011. - 271 б.
2. Митина Л.М. Психология профессионального развития учителя. М.: Изд-во «Флинта», Московский психолого-социальный институт, 2008. - 200 с
3. Owens, John D, Houston, Mike, Luebke, David, Green, Simon, Stone, John E, Phillips, James C, “GPU Computing”, *Proceedings of the IEEE*, vol.96, No.5, pp.879-899, May 2008.