**ӘОЖ 004.42**
**МҒТАР 50.05.09**
https://doi.org/10.47526/2024-1/2524-0080.09

**Н.М. ЖУНИСОВ[1], Ә.Т. БАЯЛЫ[2], Е.Т. САТЫБАЛДЫ[3]**
*[1]PhD, аға оқытушы,*
*Қожа Ахмет Ясауи атындағы Халықаралық қазақ-түрік университеті, (Қазақстан, Түркістан қ)., e-mail: nurseit.zhunissov@ayu.edu.kz*
*[2]Аға оқытушы, Қожа Ахмет Ясауи атындағы Халықаралық қазақ-түрік университеті, (Қазақстан, Түркістан қ)., e-mail: azimkhan.bayaly@ayu.edu.kz*
*[3]Студент, Қожа Ахмет Ясауи атындағы Халықаралық қазақ-түрік университеті, (Қазақстан, Түркістан қ.) e-mail: ersultan140818@gmail.com*

# PYTHON ТІЛІНІҢ КӨМЕГІМЕН ПАРАЛЛЕЛЬ БАҒДАРЛАМАЛАУДЫ ҚОЛДАНУ МҮМКІНДІКТЕРІ

**Аңдатпа.** Бұл мақалада Python бағдарламалау тілін қолдана отырып, параллельді бағдарламалық қосымшалардың дамуын зерттейді. Параллельді бағдарламалау Ақпараттық технологиялар әлемінде барған сайын маңызды бола түсуде, өйткені көп ядролы процессорлар мен таратылған есептеулер жиі кездеседі. Python әзірлеушілерге параллель қосымшаларды құруға арналған көптеген құралдар мен кітапханаларды ұсынады, соның ішінде ағындар (threads), процестер (processes), және асинхронды бағдарламалау.

Бұл тақырып Python-дағы параллельді бағдарламалау негіздерін, соның ішінде ағындар мен процестерді басқару принциптерін, қателерді өңдеуді, синхрондау механизмдерін және ресурстарды басқаруды қамтиды. Ол сонымен қатар асинхронды тапсырмаларды тиімді өңдеуге мүмкіндік беретін asyncio кітапханасын пайдаланып асинхронды бағдарламалауды қарастырады.

Сонымен қатар, бұл тақырып параллель қосымшаларды оңтайландыру және профильдеу мәселелерін көтереді, сонымен қатар үшінші тарап кітапханалары мен шеңберлерін қолдана отырып таратылған параллель бағдарламалауды зерттейді. Ол сонымен қатар параллельді бағдарламалау контекстінде тестілеу мен күйін келтірудің маңыздылығын атап көрсетеді.

Python көмегімен параллельді бағдарламалау саласындағы зерттеулер мен тәжірибелер әзірлеушілерге көп ядролы жүйелер мен үлестірілген есептеулерді тиімді қолдана алатын жоғары өнімді және тиімді қосымшаларды құруға көмектеседі.

Бұл мақалада Python тілі тәжірибесіз студенттерге параллель бағдарламалауды үйрету үшін қаншалықты қолайлы екенін қарастыратын терең зерттеу ұсынылған. Нәтижелер Python-да дәйекті бағдарламалаудан параллельге көшу кезінде оның артықшылықтарын сақтауға кедергі келтіретін кедергілер бар екенін көрсетеді.

**Кілттік сөздер.** Python, параллель, параллелизм, параллельді бағдарламалау, ақпараттық технологилар, ағындар, процестер.

**Н.М. Жунисов[1], А.Т. Баялы[2], Е.Т. Сатыбалды[3]**
*[1]PhD, старший преподаватель,*
*Международного казахско-турецкого университета имени Ходжи Ахмеда Ясави, (Казахстан, г. Туркестан), e-mail: nurseit.zhunissov@ayu.edu.kz*
*[2]Старший преподаватель, Международного казахско-турецкого университета имени Ходжи Ахмеда Ясави, (Казахстан, г. Туркестан), e-mail: azimkhan.bayaly@ayu.edu.kz*
*[3]Студент, Международного казахско-турецкого университета имени Ходжи Ахмеда Ясави, Казахстан, г. Туркестан, e-mail: ersultan140818@gmail.com*

# ВОЗМОЖНОСТИ ПРИМЕНЕНИЯ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ С ПОМОЩЬЮ ЯЗЫКА PYTHON

**Аннотация.** В этой статье исследуется разработка параллельных программных приложений с использованием языка программирования Python. Параллельное программирование становится все более важным в мире информационных технологий, поскольку многоядерные процессоры и распределенные вычисления становятся все более распространенными. Python предоставляет разработчикам множество инструментов и библиотек для создания параллельных приложений, включая потоки (threads), процессы (processes), и асинхронное программирование.

Эта тема охватывает основы параллельного программирования в Python, включая принципы управления потоками и процессами, обработку ошибок, механизмы синхронизации и управление ресурсами. Он также рассматривает асинхронное программирование с использованием библиотеки asyncio, которая позволяет эффективно обрабатывать асинхронные задачи.

Кроме того, эта тема поднимает вопросы оптимизации и профилирования параллельных приложений, а также исследует распределенное параллельное программирование с использованием сторонних библиотек и фреймворков. Он также подчеркивает важность тестирования и отладки в контексте параллельного программирования.

Исследования и эксперименты в области параллельного программирования с использованием Python помогают разработчикам создавать высокопроизводительные и эффективные приложения, которые могут эффективно использовать многоядерные системы и распределенные вычисления.

В этой статье предлагается углубленное исследование, в котором рассматривается, насколько язык Python подходит для обучения параллельному программированию неопытных студентов. Результаты показывают, что существуют препятствия, которые мешают Python сохранять свои преимущества при переходе от последовательного программирования к параллельному.

**Ключевые слова.** Python, параллель, параллелизм, параллельное программирование, информационные технологии, потоки, процессы.

**N.M. ZHunissov[1], A.T. Bayaly[2], E.T. Satybaldy[3]**

*[1]PhD, Khoja Akhmet Yassawi International Kazakh-Turkish University, Kazakhstan, Turkistan, e-mail: nurseit.zhunissov@ayu.edu.kz*

*[2]Senior Lecturer, Khoja Akhmet Yassawi International Kazakh-Turkish University, Kazakhstan, Turkistan, e-mail: azimkhan.bayaly@ayu.edu.kz*

*[3]Student, Khoja Akhmet Yassawi International Kazakh-Turkish University, Kazakhstan, Turkistan, e-mail: ersultan140818@gmail.com*

# THE POSSIBILITIES OF USING PARALLEL PROGRAMMING USING PYTHON

**Annotation.** This article explores the development of parallel software applications using the Python programming language. Parallel programming is becoming increasingly important in the information technology world as multi-core processors and distributed computing become more common. Python provides developers with a variety of tools and libraries for creating parallel applications, including threads, processes, and asynchronous programming.

This topic covers the basics of parallel programming in Python, including the principles of thread and process management, error handling, synchronization mechanisms, and resource management. He also considers asynchronous programming using the asyncio library, which allows you to efficiently handle asynchronous tasks.

In addition, this topic raises issues of optimization and profiling of parallel applications, as well as explores distributed parallel programming using third-party libraries and frameworks. He also emphasizes the importance of testing and debugging in the context of parallel programming.

Research and experiments in parallel programming using Python help developers create high-performance and efficient applications that can effectively use multi-core systems and distributed computing.

This article offers an in-depth study that examines how Python is suitable for teaching parallel programming to inexperienced students. The results show that there are obstacles that prevent Python from maintaining its advantages in the transition from sequential programming to parallel.

**Keywords.** Python, parallel, parallelism, parallel programming, information technology, flows, processes.

**Кіріспе**

Python бағдарламалау тілі жаңадан бастаушыларға дәйекті бағдарламалауды үйрету үшін қолайлы тіл ретінде танымал бола бастады. Python синтаксисі таза, жеңіл және бәріне түсінікті. Сонымен қатар, бұл императивті, функционалды және объектіге бағытталған бірнеше бағдарламалау парадигмаларын қолдайтын жоғары деңгейлі бағдарламалау тілі. Сондықтан да, Python параллельді бағдарламалау парадигмаларын үйрету үшін қолайлы тіл деп атауға болатыны анық.

Python көмегімен параллельді бағдарламалық жасақтама жасауды үйрену көп тапсырмалы және көп ағынды қосымшаларды құру дағдыларын жетілдіргісі келетін әзірлеушілер үшін пайдалы болуы мүмкін. Параллельді бағдарламалау бір уақытта бірнеше тапсырмаларды орындауға мүмкіндік береді, бұл бағдарламалардың өнімділігін едәуір арттырады және тапсырмалардың орындалу уақытын қысқартады.

Python тілі - жетілдірілген интерпретацияланған бағдарламалау тілі [1]. Ол бірнеше бағдарламалау платформалары мен парадигмаларын қолдайды. Тілдің сипаттамалары келесідей: оқу мен қолданудың қарапайымдылығы, кеңеюі және көптеген бағдарламалық кітапханалар оқытушылар мен бағдарламалық жасақтама жасаушыларды қызықтырды. Python бүгінде академияда дәйекті бағдарламалаудың ең танымал кіріспе тілі болып табылады [2]. Соңғы жылдары бұл бағдарламалық жасақтама жасаушылар арасында ең танымал бағдарламалау тілі болды. Бағдарламалау тілдерінің танымалдылығының көрсеткіші TIOBE Software 2022 жылдың тамызына арналған ең танымал бағдарламалау тілдерінің рейтингін ұсынды. Өткен жылмен салыстырғанда Python танымалдығы 3,56% - арттырып, 15,42% көрсеткішімен екінші орыннан бірінші орынға көшті (Сурет-1). Бұл рейтингтің барлық уақытында осы бағдарламалау тілінің танымалдылығының ең жоғары көрсеткіші. Ең төменгісі 2003 жылы (0,97%) Python рейтингте 13-ші орынға ие болған кезде тіркелді[3].

| Aug 2022 | Aug 2021 | Change | Programming Language | Ratings | Change |
|---|---|---|---|---|---|
| 1 | 2 | ↑ | Python | 15.42% | +3.56% |
| 2 | 1 | ↓ | C | 14.59% | +2.03% |
| 3 | 3 | | Java | 12.40% | +1.96% |
| 4 | 4 | | C++ | 10.17% | +2.81% |
| 5 | 5 | | C# | 5.59% | +0.45% |
| 6 | 6 | | Visual Basic | 4.99% | +0.33% |
| 7 | 7 | | JavaScript | 2.33% | -0.61% |
| 8 | 9 | ↑ | Assembly language | 2.17% | +0.14% |
| 9 | 10 | ↑ | SQL | 1.70% | +0.23% |
| 10 | 8 | ↓ | PHP | 1.39% | -0.80% |

Сурет-1. Python тілінің танымалдығының көрсеткіші.

Егер бұл статистикалар бүгінгі таңда танымал бағдарламалауды оқыту бойынша оқу орындарының философиясын көрсетсе, онда бұл оқу орындарында жақын арада Python көмегімен параллельді бағдарламалауды оқытуды таңдайды деп болжау қате болмайды. Сериялық бағдарламалауды үйренуге жарамды тіл параллельді бағдарламалауды үйрену үшін қолайлы таңдау болып табылады деген болжам қате болуы мүмкін. Бұл зерттеуде біз Python параллельді бағдарламалауды бастаушыларға үйрету үшін қаншалықты қолайлы екенін қарастырамыз. Бұл зерттеуде қолданылған әдістеме қазіргі уақытта бағдарламалық жасақтама жасаушыларға қол жетімді әдебиеттерді, әдістер мен құралдарды жан-жақты шолуды және оларды әртүрлі алгоритмдерді қолдана отырып тәжірибеде зерттеуді қамтыды.

Тіл параллельді бағдарламалаудың кіріспе курстарын оқытуға жарамды, егер ол келесі қажетті сипаттамаларға ие болса:

- Түсіну және кодтау үшін жоғары деңгейлі абстракцияны және қарапайым синтаксисті қолдайды
- Орнату оңай
- Әр түрлі аппараттық және бағдарламалық платформалар қолдайды
- Академиялық мақсаттар үшін тегін
- Параллельді бағдарламалаудың үш негізгі парадигмасын қолдайды: ортақ жад (көп ағынды негізінде), үлестірілген жад (хабарлама жіберу негізінде) және гетерогенді бағдарламалау
- Бейнелеу профильдеу және күйін келтіру құралдарымен қолдау көрсетіледі
- Академиялық және өндірістік стандарттармен қамтамасыз етілген
- Масштабталуымен ерекшеленетін қарапайым алгоритмдерді көрсетудің қарапайымдылығы

Python-дағы бағдарламалық жасақтама жасаушыларға параллелизмді немесе Python-дағы Даму орталары үшін арнайы оңтайландыру әдістерін қолдану арқылы қосымшалардың өнімділігін арттыру үшін қол жетімді негізгі әдістерді, кітапханаларды және құралдарды қарастырады[4].

Бұл мақалада біз үш шешімге тоқталамыз: Python ағындық модулі, көп процессорлы модуль және Numba. Біз бұл шешімдерді таңдаймыз, өйткені олар параллельді бағдарламалаудың үш негізгі парадигмасын қамтиды (көп ағынды, хабарлама жіберу және гетерогенді бағдарламалау), сонымен қатар олар басқа шешімдердің күшті және әлсіз

жақтарын көрсетеді.

Мұндай оқытудың кейбір негізгі аспектілері мен ұсыныстары:

Python негіздерін үйрену:

Параллельді бағдарламалауды бастамас бұрын, Python негіздерін, соның ішінде айнымалылармен, деректер құрылымдарымен, мүмкіндіктермен, сыныптармен және модульдермен жұмыс істеуді жақсы білетіндігіңізге көз жеткізіңіз.

Параллельді бағдарламалау негіздері:

Көп тапсырма және көп ағынды ұғымдарды түсіну маңызды. Көп тапсырма, көп ағынды және көп процесс сияқты параллель парадигмалардың әртүрлі түрлерін зерттеңіз.

Python-да көп тапсырма:

Python асинхронды бағдарламалауға жарамды asyncio және параллель процестерді іске қосуға мүмкіндік беретін multiprocessing сияқты көп тапсырмалы модульдерді ұсынады. Осы модульдерді және олардың қолданылуын зерттеңіз.

Python-да көп ағынды:

Python-да көп ағындармен жұмыс істеу үшін threading модулін пайдалануға болады. Ағындарды қалай құруға және басқаруға болатындығын біліңіз

**Зерттеу әдістері**

Бүгінгі таңда көп ағынды бағдарламалау көп ядролы процессорлардың артықшылықтарын пайдаланатын бағдарламалық қосымшаларда параллелизмді қолдану үшін ең көп қолданылатын параллельді бағдарламалау парадигмасы болып табылады. Мұндай бағдарламалау техникасы пайдалану қиындықтарына және шешілуі керек мәселелерге әкелуі мүмкін [5]. Дегенмен, көп ағынды механизмі бар заманауи қолданбалар барлық жерде кең таралған және күннен-күнге танымал болып келеді. Көп ағынды көптеген операциялық жүйелер, сондай-ақ параллельді бағдарламалау модельдері кеңінен қолданатындықтан және қолдайтындықтан, бұл параллельді бағдарламалаудың кіріспе курстарында оқытылатын алғашқы параллельді бағдарламалау парадигмасы.

Ағындық бағдарламалау жалпы кеңістіктегі адрестеуді қолданатын ағындар арасындағы байланыс әдісін қолдайды. Бірнеше ағындар деректерді және көптеген ресурстарды бөлісе алады. Процестерді қолданудан гөрі ағындардың артықшылығы-өнімділік, өйткені ағындар арасындағы контекстті ауыстыру процестер арасындағы контекстті ауыстырудан әлдеқайда оңай.

Python-да ағындарды басқару стандартты Python кітапханасы ұсынатын threading пакеті арқылы жүзеге асырылады.

**`multiprocessing`** мультипроцессор-бұл threading модуліне ұқсас API көмегімен процестерді іске қосуды қолдайтын пакет. Мультипроцессорлық пакет жергілікті және қашықтағы параллелизмді ұсынады, ағындардың орнына ішкі процестерді қолдану арқылы аудармашының жаһандық құлпын тиімді айналып өтеді. Осының арқасында мультипроцессорлық модуль бағдарламашыға берілген машинада бірнеше процессорларды толығымен пайдалануға мүмкіндік береді. Ол Unix және Windows жүйелерінде жұмыс істейді.

Мультипроцессорлық модуль сонымен қатар көп ағынды модульде аналогтары жоқ API интерфейстерін енгізеді. Мұның жарқын мысалы-pool нысаны, ол бірнеше кіріс мәндері бойынша функцияны орындауды параллельдеудің ыңғайлы құралын ұсынады, кіріс деректерін процестерге бөледі (деректер параллелизмі). Келесі мысал модульде осындай функцияларды анықтаудың кең таралған тәжірибесін көрсетеді, осылайша балалар процестері осы модульді сәтті импорттай алады[6].

Мысалы **`Pool`** қолданатын мәліметтер параллелизмінің негізгі функциясы:

```
from multiprocessing import Pool
```

```
def f(x):
    return x*x

if __name__ == '__main__':
    p = Pool(5)
    print(p.map(f, [1, 2, 3]))
```

Көп процессорлы өңдеу кезінде процестер process нысанын құру және одан кейін оның start () әдісін шақыру арқылы іске қосылады. Процесс көп ағынды API-ге сәйкес келеді. Мысалы Мультипроцессорлық бағдарламаның маңызды емес функцисы:

```
from multiprocessing import Process

def f(name):
    print 'hello', name

if __name__ == '__main__':
    p = Process(target=f, args=('bob',))
    p.start()
    p.join()
```

Python тілін қолдана отырып, параллельді бағдарламалауды дамытуды үйрену тиімді және көп тапсырмалы қосымшалар жасағысы келетін әзірлеушілер үшін маңызды дағды болуы мүмкін. Міне, осы дағдыны үйренуге көмектесетін қадамдар:

Python негіздері: параллельді бағдарламалауға кіріспес бұрын, синтаксисті, деректер құрылымын, функцияларды және модульдерді білуді қоса алғанда, Python негіздерінің берік екендігіне көз жеткізіңіз[7,8].

Ағындар мен процестерді зерттеңіз: Python-дағы ағындар мен процестер арасындағы айырмашылықты түсіну маңызды. Python ағындармен жұмыс істеу үшін threading және процестермен жұмыс істеу үшін multiprocessing модульдерін ұсынады.

Асинхронды бағдарламалау: asyncio кітапханасын пайдаланып асинхронды бағдарламалауды үйреніңіз. Бұл көптеген операцияларды бұғаттаусыз тиімді өңдейтін асинхронды қосымшаларды құруға мүмкіндік береді.

GIL (Global Interpreter Lock): Python-да бір процесте бірнеше ағындардың орындалуын шектейтін GIL бар екенін түсіну пайдалы болуы мүмкін. Бұл дегеніміз, тапсырмаларды қатар орындау үшін кейбір жағдайларда ағындарды емес, процестерді қолданған дұрыс.

Кітапханалар мен құрылымдар: Python-да параллель қосымшаларды әзірлеуді жеңілдететін үшінші тарап кітапханалары мен құрылымдарын зерттеңіз. Мысалы, concurrent.futures, asyncio, Celery, Dask және басқалары.

Көп тапсырмалы үлгілерді үйреніңіз: MapReduce, Producer-Consumer және т.б. сияқты көп тапсырмалы үлгілерді үйреніңіз. Олар сізге тапсырмаларды қатар орындауды қалай ұйымдастыруға болатындығын түсінуге көмектеседі[9,10].

Қателерді өңдеу және қауіпсіздік: параллель қосымшаларда синхрондау мен қателерді өңдеуді ескеру маңызды. Деректер жарысының алдын алу және ерекшеліктерді басқару жолдарын зерттеңіз.

Параллельді қосымшаларды жобалау: параллельді қосымшаларды жобалауды үйреніңіз. Қолданбаның қай бөліктері параллель болуы мүмкін екенін және олардың бір-

бірімен қалай әрекеттесетінін анықтаңыз.

Тәжірибе және жобалар: параллель бағдарламалауды меңгеруде тәжірибе маңызды рөл атқарады. Алған біліміңізді қолданатын өз жобаларыңызды жасаңыз.

Оқу және оқыту: кітаптарды, онлайн курстарды оқу, вебинарларға қатысу және тәжірибелі әзірлеушілермен байланыс параллельді бағдарламалау туралы біліміңізді тереңдетуге көмектеседі.

Тәжірибелер және оңтайландыру: кодты оңтайландыру үшін профильдер мен құралдармен жұмыс істеу параллель қолданбалардың өнімділігін жақсартуға мүмкіндік береді.

Ынтымақтастық және тәжірибе алмасу: тәжірибе алмасуға және басқа әзірлеушілерден кеңес алуға болатын Python қауымдастықтары мен форумдарына қосылыңыз.

Python-да параллельді бағдарламалау қиын болуы мүмкін, бірақ тәжірибе мен ең жақсы тәжірибелерді үйрену арқылы сіз осы тілде тиімді және көп тапсырмалы қосымшалар жасай аласыз[11].

**Талдау мен нәтижелер**

Python-да мультипроцессорлық өңдеу жасау үшін "multiprocessing" модулін пайдалануға болады. Бірнеше процессінің біріні кездейсоқ, бірнеше нысаналы процестерде анықтау мақсатында мультипроцессорлық өңдеу пайдаланылады. Қарастырып көрсек:

1. multiprocessing модулін импорттау: Өз аппараттық түрінше Python-да мультипроцессорлық өңдеу жасау үшін "multiprocessing" модулін импорттау керек.

```
python
import multiprocessing
```

2. Процестерді жасау: "multiprocessing" модулі арқылы жаңа процесстерді жасау мүмкін. Кейбір тақырыптарды басқару үшін "Process" класын пайдалануыңыз керек.

```
python
def my_function(x):
    result = x * x
    print(f"Болмаса, result: {result}")

if __name__ == "__main__":
    # Өңдеуді жасау үшін жаңа процесстерді жасау
    process1 = multiprocessing.Process(target=my_function, args=(5,))
    process2 = multiprocessing.Process(target=my_function, args=(10,))

    # Процестерді іске асыру
    process1.start()
    process2.start()

    # Процестерді күтіп жату
    process1.join()
    process2.join()

    print("Барлығы орындалды.")
```

Өңдеу сәтті орындалды. "if name == "main":", "start()" және "join()" қолданбағын пайдалануыңыз керек, сондықтан өзінізге жариялауға кеңес берілген жердег пайда бермейді.

3. Процесстердің мәліметтерін біріктіру: Бірнеше процестерді жұмыс жасау үшін бір жаттығу қажет болмаса, процесстердің алдындағы жаңа өрістер арқылы мәліметтерді біріктіру керек. Мысалы, "multiprocessing.Queue" пайдаланылуы мүмкін.

python

```
import multiprocessing

def worker(q, x):
    result = x * x
    q.put(result)

if __name__ == "__main__":
    # Queue жасау
    result_queue = multiprocessing.Queue()

    # Процестерді жасау
    process1 = multiprocessing.Process(target=worker, args=(result_queue, 5))
    process2 = multiprocessing.Process(target=worker, args=(result_queue, 10))

    # Процестерді іске асыру
    process1.start()
    process2.start()

    # Процестерді күтіп жату
    process1.join()
    process2.join()

    # Мәліметтерді алу
    result1 = result_queue.get()
    result2 = result_queue.get()

    print(f"Болмаса, result1: {result1}")
    print(f"Болмаса, result2: {result2}")
```

Бұл Процесстердің мәліметтерін біріктіру мысалында, "Queue" арқылы процестерден мәліметтерді біріктіру мүмкіндіктерін берді[12].

Жеткілікті көптілеу, басқару және алдау нысандарын басқару үшін "multiprocessing" модулін пайдалануыңыз керек.

**Қорытынды**

Python тілін қолдана отырып, параллельді бағдарламалық қосымшаларды әзірлеу ақпараттық технологиялар әлеміндегі маңызды және өзекті аспект болып табылады. Көп ядролы процессорлардың үздіксіз өсуімен және үлкен көлемдегі деректерді өңдеу қажеттілігімен тиімді және көп тапсырмалы қосымшаларды құра білу әзірлеушілер үшін негізгі дағдыға айналады.

Бұл тақырып әзірлеушілерге Python-да параллельді бағдарламалаудың әртүрлі әдістерін, соның ішінде ағындарды, процестерді және асинхронды бағдарламалауды меңгеруге мүмкіндік береді. Әрбір әдістің күшті және әлсіз жақтары бар екенін түсіну маңызды және әдісті таңдау қолданбаның нақты талаптарына байланысты.

Сонымен қатар, тақырып параллельді қосымшаларды мұқият тестілеу, күйін келтіру және оңтайландыру қажеттілігін көрсетеді. Параллельді бағдарламалардағы қателерді анықтау және түзету бір ағынды қолданбаларға қарағанда қиынырақ болуы мүмкін, сондықтан сауатты тестілеу және кодты профильдеу дамудың ажырамас бөлігіне айналады.

Параллельді бағдарламалау үшін үшінші тарап кітапханалары мен құрылымдарын пайдалану қосымшаларды әзірлеуді және кеңейтуді едәуір жеңілдетеді.

Қорытындылай келе, Python көмегімен параллельді қосымшаларды әзірлеу әзірлеушілерге жоғары өнімді және тиімді бағдарламаларды құрудың көптеген құралдары мен мүмкіндіктерін ұсынады. Бұл дағды қазіргі ақпараттық технологиялар әлемінде барған сайын құнды бола түсуде және көп тапсырмалы және таратылған ортада жұмыс істей алатын қуатты және жылдам қосымшаларды құруға есік ашуы мүмкін.

## ПАЙДАЛАНЫЛҒАН ӘДЕБИЕТТЕР ТІЗІМІ


1. Jan Palach. Parallel Programming with Python. Packt Publishing. July 14, 2014. 124 pages
2. Лупин, С.А. Технологии параллельного программирования / С.А. Лупин. - М.: Форум, 2018.-**402**с.
3. Тормасов, А.Г. Параллельное программирование многопоточных систем с разделяемой памятью / А.Г. Тормасов. - М.: Физматкнига,**2020**.-**296с.**
4. L. Dalcin and Y.-L. L. Fang, *mpi4py: Status Update After 12 Years of Development*, Computing in Science & Engineering, 23(4):47-54, 2021. https://doi.org/10.1109/MCSE.2021.3083216
5. L. Dalcin, P. Kler, R. Paz, and A. Cosimo, *Parallel Distributed Computing using Python*, Advances in Water Resources, 34(9):1124-1139, 2011. https://doi.org/10.1016/j.advwatres.2011.04.013
6. L. Dalcin, R. Paz, M. Storti, and J. D'Elia, *MPI for Python: performance improvements and MPI-2 extensions*, Journal of Parallel and Distributed Computing, 68(5):655-662, 2008. https://doi.org/10.1016/j.jpdc.2007.09.005
7. L. Dalcin, R. Paz, and M. Storti, *MPI for Python*, Journal of Parallel and Distributed Computing, 65(9):1108-1115, 2005. https://doi.org/10.1016/j.jpdc.2005.03.010
8. Лагунова А. Д. Параллельные вычисления как способ повышения эффективности алгоритма летучих мышей. // Сборник статей IX Международной научно-практической конференции «Инновационное развитие науки и образования».— 2020.— С. 78–87
9. Лагунова А.Д., Назаров Д. А. Параллельный алгоритм решения задачи оптимального параметрического синтеза на основе метода сеток. // Труды Международного симпозиума «Надежность и качество».— 2018.— Т.1.— С. 255–258
10. Andrews, G. R.Fundamentals of multithreaded, parallel and distributed programming, Moscow: Williams, 2003.
11. Златопольский Д.М. Основы программирования на языке Python. - М.: ДМК Пресс, 2017. – 284 с.
12. *Шелудько, В. М.* Основы программирования на языке высокого уровня Python: учебное пособие / В. М. Шелудько. – Ростов-на-Дону, Таганрог: Издательство Южного федерального университета, 2017. – 146 с.


## REFERENCES


1. Jan Palach. Parallel Programming with Python. Packt Publishing. July 14, 2014. 124 pages
2. Lupin, S.A. Tekhnologii parallel'nogo programmirovaniya / S.A. Lupin. - M.: Forum, 2018. - 402 c.

3. Tormasov, A.G. Parallel'noe programmirovanie mnogopotochnyh sistem s razdelyaemoj pamyat'yu / A.G. Tormasov. - M.: Fizmatkniga, 2020. - 296 c.
4. L. Dalcin and Y.-L. L. Fang, mpi4py: Status Update After 12 Years of Development, Computing in Science & Engineering, 23(4):47-54, 2021. https://doi.org/10.1109/MCSE.2021.3083216
5. L. Dalcin, P. Kler, R. Paz, and A. Cosimo, Parallel Distributed Computing using Python, Advances in Water Resources, 34(9):1124-1139, 2011. https://doi.org/10.1016/j.advwatres.2011.04.013
6. L. Dalcin, R. Paz, M. Storti, and J. D'Elia, MPI for Python: performance improvements and MPI-2 extensions, Journal of Parallel and Distributed Computing, 68(5):655-662, 2008. https://doi.org/10.1016/j.jpdc.2007.09.005
7. L. Dalcin, R. Paz, and M. Storti, MPI for Python, Journal of Parallel and Distributed Computing, 65(9):1108-1115, 2005. https://doi.org/10.1016/j.jpdc.2005.03.010
8. Lagunova A. D. Parallel'nye vychisleniya kak sposob povysheniya effektivnosti algoritma letuchih myshej (BA) / A. D. Lagunova // Sbornik statej IX Mezhdunarodnoj nauchno-prakticheskoj konferencii «Innovacionnoe razvitie nauki i obrazovaniya».— 2020.— S. 78–87
9. Lagunova A.D., Nazarov D. A. Parallel'nyj algoritm resheniya zadachi optimal'nogo parametricheskogo sinteza na osnove metoda setok // Trudy Mezhdunarodnogo simpoziuma «Nadezhnost' i kachestvo».— 2018.— T. 1.— S. 255–258
10. Andrews, G. R.Fundamentals of multithreaded, parallel and distributed programming, Moscow: Williams, 2003.
11. Zlatopol'skij D.M. Osnovy programmirovaniya na yazyke Python. – M.: DMK Press, 2017. – 284 s.
12. SHelud'ko, V. M. Osnovy programmirovaniya na yazyke vysokogo urovnya Python: uchebnoe posobie / V. M. SHelud'ko. – Rostov-na-Donu, Taganrog: Izdatel'stvo YUzhnogo federal'nogo universiteta, 2017. – 146 c.