5 - дәріс. Қарапайым параллельді алгоритмдер.

Жоспар:

- Параллельді есептеулердің негізгі компоненттері
- Қарапайым параллельді алгоритмдерге мысалдар
- Параллельді бағдарламалаудағы қиындықтар

Бізді қоршаған әлем – әрекеттердің бір мезгілде орындалатын әлемі. Адам миының екі жартысы әртүрлі міндеттерді орындайды, автомобильдер бір уақытта көшелерде жүреді. Компьютерлік әлем де осы заңдылыққа бағынады. Бір процессордың жылдамдығының өсуі шектеулерге тап болған кезде, есептеулерді жеделдетудің негізгі жолы – оларды бір уақытта орындау, яғни **параллелизм**.

Бүгінгі лекцияда біз параллельді есептеулердің негізін және ең қарапайым параллельді алгоритмдерді қарастырамыз.

1. Параллельді есептеулердің негізгі компоненттері

- **Task (тапсырма):** Орындалатын жеке жұмыс бірлігі.
- **Процессор (Processor):** Тапсырманы орындауға арналған физикалық немесе логикалық құрылғы.
- **Коммуникация (Communication):** Процессорлар арасындағы дерек алмасу.
- **Синхронизация (Synchronization):** Әртүрлі процессорлардағы тапсырмалардың орындалу ретін басқару.

Параллельдік моделі: Fork-Join
Бұл ең интуитивті модель.

- **Fork (Тарату):** Негізгі бағдарлама (негізгі процесс) бірнеше параллельді ішкі тапсырмаларды (процестерді) бастайды.
- **Join (Қосу):** Негізгі бағдарлама барлық ішкі тапсырмалар аяқталғанша күтеді, содан кейін нәтижелерді біріктіреді.

2. Қарапайым параллельді алгоритмдерге мысалдар

2.1. Массив элементтерінің қосындысы (Array Sum)

- **Мәселе:** n өлшемді массивтің барлық элементтерінің қосындысын табу.
- **Тізбекті шешім:** Цикл арқылы қосу. Күрделілігі: O(n).
- **Параллельді шешім (Ағаш әдісі):**

1. **Fork:** Массивті p бөлікке бөлеміз. Әр процессор өз бөлігінің қосындысын есептейді.
2. **Join:** Барлық процессорлардың жеке қосындыларын қосып, жалпы қосындыны аламыз. Бұл қосу процесі екілік ағаш түрінде жүзеге асырылуы мүмкін.

Мысал (8 элемент, 4 процессор):

text
Массив: [2, 4, 1, 7, 5, 3, 6, 2]

1. Әр процессор өз бөлігін қосады:
   P0: 2+4 = 6
   P1: 1+7 = 8
   P2: 5+3 = 8
   P3: 6+2 = 8

2. Ағаш тәсілімен қосындыларды біріктіреміз:
   (P0+P1): 6+8 = 14
   (P2+P3): 8+8 = 16
   (14 + 16) = 30 // Жалпы қосынды

- **Талдау:** Бірінші кезең O(n/p) уақыт алады. Екінші кезең (ағашты біріктіру) O(log p) уақыт алады. Жалпы уақыт: O(n/p + log p). p << n болғанда, жеделдету шамамен p-ге жақын болады.

2.2. Параллельді іздеу (Parallel Search)

- **Мәселе:** Массивтегі белгілі бір кілтті элементті табу.
- **Тізбекті шешім:** Сызықтық іздеу. Күрделілігі: O(n).
- **Параллельді шешім:**

1. **Fork:** Массивті p бөлікке бөлеміз. Әр процессор өз бөлігінде кілтті іздейді.
2. **Join:** Кез келген процессор элементті тапса, ол бәріне тапқанын хабарлайды (немесе негізгі бағдарламаға жібереді). Басқа процессорлар жұмысын тоқтатады.

- **Талдау:** Ең жақсы жағдайда, элемент бірінші процессордың бірінші позициясында болуы мүмкін. Сонда уақыт O(1). Ең нашар жағдайда, элемент соңғы процессордың соңғы позициясында болуы мүмкін. Сонда уақыт O(n/p). Бұл алгоритмді жеделдету элементтің орналасуына тәуелді.

2.3. Көпіршік сұрыптауға қарапайым параллельді тәсіл (Bubble Sort Variant)

- **Мәселе:** Массивті өсу ретімен реттеу.
- **Тізбекті шешім: Көпіршік сұрыптауға** . Күрделілігі: $O(n^2)$.
- **Параллельді шешім (Күрделі жұп-тақ сорттау - Odd-Even Transposition Sort):**
  Бұл алгоритм n процессор болғанда жұмыс істейді және әрекеттерді екі кезеңге бөледі:

  - **Тақ фаза:** (i=1,3,5,...) тақ индекстердегі элементтерді көршілес жұп индексті элементтермен (i және i+1) салыстырып, қажет болса орындарын ауыстырады.
  - **Жұп фаза:** (i=2,4,6,...) жұп индекстердегі элементтерді көршілес тақ индексті элементтермен (i және i+1) салыстырып, қажет болса орындарын ауыстырады.

  Бұл екі фаза n/2 рет қайталанады.

  Мысал (4 элемент, 4 процессор):

  text
  Бастапқы күй: [4, 2, 3, 1]

1-ші тақ фаза: (4<->2), (3<->1) -> [2, 4, 1, 3]
1-ші жұп фаза: (4<->1) -> [2, 1, 4, 3]
2-ші тақ фаза: (2<->1), (4<->3) -> [1, 2, 3, 4]
2-ші жұп фаза: (2<->3) өзгеріс жоқ -> [1, 2, 3, 4] // Дайын

- **Талдау:** Алгоритм n параллель салыстыру операциясын орындайды және оны n/2 рет қайталайды. Сондықтан параллель уақыт күрделілігі O(n). Бұл тізбектік $O(n^2)$ алгоритмінен жақсы.

3. Параллельді бағдарламалаудағы қиындықтар

1. **Коммуникациялық шығындар:** Процессорлар арасындағы дерек алмасу әрқашан есептеуден баяу.
2. **Жүктемені теңестіру (Load Balancing):** Егер бір процессордағы тапсырма басқаларынан әлдеқайда ауыр болса, онда ол барлық жүйенің күтуына себеп болады.
3. **Синхронизация:** Процессорлардың бір-бірінің аяқталуын дұрыс күтуі керек, әйтпесе қате нәтижелер пайда болады.
4. **Бөлінген деректерге қатынасу (Race Condition):** Екі процессор бір деректерді бір уақытта оқып/жазғысы келсе, нәтиже бекітілмейді.

Қорытынды

Параллельді алгоритмдер – бұл заманауи есептеулердің негізі. Біз бүгін қарастырған алгоритмдер (қосынды, іздеу, қарапайым сорттау) өте қарапайым, бірақ олар параллельді ойлау мен талдаудың негізгі принциптерін түсінуге көмектеседі. Күрделі есептерді шешу үшін (мысалы, матрицаны көбейту, графтардағы алгоритмдер) осы негізгі идеяларды пайдаланатын әлдеқайма күрделі әдістер қолданылады. Параллельдік – бұл жылдамдықты арттырудың күшті құралы, бірақ оны тиімді пайдалану үшін мұқият жобалау және жүктемені теңестіру қажет.