9 - дәріс. OpenMP: процессорды байланыстыру. OpenMP конструкциялары. Деректер жарысы.

Жоспар:

1. Бағдарламалау тіліне арналған OpenMP кеңейтімдері
2. OpenMP конструкциялары
3. Деректер жарысы (Data Race condition)

Бағдарламалау тіліне арналған OpenMP кеңейтімдері

1) Параллель басқару құрылымдары
   Бағдарламадағы бақылау ағынын басқарады
2) Жұмысты бөлу (Work shairing)
   Тапсырмаларды ағындар арасында бөледі
3) Деректерді өңдеу
   Деректер ауқымының айнымалылары
4) Синхронизациялау
   Ағындардың жұмысын үйлестіреді
5) Runtime функциялары мен орта айнымалылары
   Орындалу орталары

OpenMP конструкциялары

OpenMP конструкциясы оған әсер ететін орындалатын кодпен бірге OpenMP директивасынан тұрады.

OpenMP конструкциялары:

- Параллельді аймақ
  #pragma omp parallel
- Жұмысты бірлесіп орындау
  #pragma omp for, #pragma omp sections, #pragma omp single
- Мастер
- #pragma omp master
- Синхрондау
  #pragma omp barrier, #pragma omp critical, #pragma omp atomic

Параллельді аймақ

Әдетте OpenMP пайдаланатын көп ағынды бағдарлама үшін пайда болатын бірінші OpenMP конструкциясы – параллельді аймақ.

- Параллельді аймақ бағдарламаның параллель орындалатын бөлігін анықтайды.
- Байланысты параллель аймақты орындау үшін ағындар командасы құрылады.
- Аймақтағы параллель тапсырма әр ағынға таратылады.
- Параллель аймақтың соңында барлық ағындарды аймақ ішіндегі есептеу аяқталғанша күтуге мәжбүрлейтін болжамды кедергі бар. Басқа ағындар осы жерге жетпейінше ешбір ағын ілгерілей алмайды.

Негізгі синтаксисі келесідей:

C/C++ үшін	***Fortran үшін***

#pragma omp parallel [опция, опция]
{
  // Бағдарлама денесі
} // параллель аймақтың соңы
	!$omp parallel [опция[[,] опция]. . .]
	  // Бағдарлама денесі
	!$omp end parallel // параллель аймақтың соңы

Опция - арнайы атрибутты көрсетуге мүмкіндік беретін OpenMP директивасына қосылған нұсқау.

OpenMP параллель аймығы үшін қол жетімді опциялар:

- if
- private
- firstprivate
- default
- shared
- copyin
- reduction
- num_threads

if - if опиясы параллель аймақтың қолданылатынын немесе қолданылмайтынын анықтауға көмектеседі. Параллельді құрылым пайдаланылмайтын екі жағдай бар: егер осы тармақтағы скаляр өрнек жалған(FALSE) деп бағаланса немесе сол уақытта басқа параллель аймақ белсенді болса.

private - private опциясынан кейін параллель аймақтағы әрбір ағын үшін бөлек жасалатын айнымалылар тізімі жалғасады. Жеке деректердің мәндері белгілі бір конструкцияға кіру және одан шығу кезінде анықталмаған күйде болады.

firstprivate – private опциясындағыдай параллель аймақтағы әрбір ағын үшін бөлек жасалған айнымалылар тізімі. Бір айырмашылығы, деректер мәні параллель аймақтың басында анықталмаған күйде емес, параллель аймақ енгізілген кезде болған мәнге инициализацияланады.

default - ортақ пайдалану күйі анық анықталмаған кез келген айнымалы мәндердің ортақ пайдалану күйін орнатады.

shared - барлық ағындар арасында ортақ пайдаланылатын айнымалыларды тізімдеу үшін пайдалануға болады. Бұл әдетте қажет емес, себебі айнымалылардың көпшілігі ортақ пайдаланылады.

copyin - тізімдегі айнымалы мәндердің негізгі ағыннан топтағы барлық басқа ағындарға көшірілуін орындайды. Тізімдегі айнымалылар жеке болуы керек, себебі олар ортақ пайдаланылса, көшіру мағынасыз болар еді.

reduction - параллель аймақтың соңында қысқарту әрекеті орындалатын айнымалылардың тізімін береді. Опция қысқартуға арналған операторды да көрсетеді. Әрбір қысқарту айнымалысының жеке көшірмесі әрбір ағын үшін жасалады. Айнымалы мәндердің жеке көшірмелері ағындар орындалғанда жаңартылады, содан кейін барлық ағындардың жеке көшірмелері параллель аймақтың соңында біріктіріледі.

Жұмысты бірлесіп орындау.

Конструкция	C/C++	Fortran
Цикл (Loop)	#pragma omp for	!$omp do
Бөлімдер (Sections)	#pragma omp sections	!$omp sections
Жеке (Single)	#pragma omp single	!$omp single

Цикл конструкциясы циклдің итерацияларының параллель орындалуына себеп болады. Опциялары:

- private
- firstprivate
- lastprivate
- reduction
- ordered
- schedule
- collapse
- nowait

Private, firstprivate, reduction опциялары параллель аймақтағыдай әрекет етеді, тек олардың жұмысының ауқымы параллель аймақ емес, олар өзгертетін OpenMP циклі болып табылады. Атап айтқанда, егер параллель аймақ бірнеше тізбекті конструкцияларды қамтыса, айнымалы мәндерді әртүрлі циклдарда басқаша өңделуі мүмкін.

Lastprivate опцтясы *private* опциясына ұқсайды, өйткені ол циклде жұмыс істейтін әрбір ағында жеке даналары болуы керек айнымалыларды тізімдейді. Сонымен қатар, онда циклдің соңғы итерациясынан кейінгі айнымалының мәні цикл аяқталғаннан кейін қол жетімді болады. Бұл анықтамада соңғы итерация код ретімен орындалғанда соңғы орындалатын иерацияны білдіреді.

Schedule опциясының төрт түрлі мән болуы мүмкін:

- static - статикалық жоспарлау цикл итерацияларының ағындар бойынша ағын нөмірлеріне сәйкес айналым ретімен бөлінеді.
- dynamic - динамикалық жоспарлау ағындар бос болған кезде бөліктердің ағындарға тағайындалатынын білдіреді.
- guided - басқарылатын жоспарлау сәл күрделірек. Бұл динамикалық жоспарлауға ұқсас, өйткені итерациялық бөліктер қол жетімді болған кезде ағындарға тағайындалады. Бастапқы бөлік өлшемі қатысушы ағындар санына бөлінген қалған итерациялар санына пропорционал.
- runtime - орындау уақытын жоспарлау OMP_SCHEDULE ортасының айнымалы мәнін пайдалану арқылы көрсетілген түр мен бөлік өлшемі мәндерімен басқарылады. Егер бұл айнымалы мәні орнатылмаған болса, әрекет орындау арқылы анықталады.

Collapse опциясы кірістірілген циклдегі қанша циклды (n) бір үлкен циклге жиып, кесте тармағына сәйкес бөлу керектігін көрсетеді.

Әрбір цикл конструкциясының соңында жасырын тосқауыл бар. Кез келген басқа код орындалмас бұрын барлық ағындар сол жерде синхрондалады. Егер *nowait* сөйлемі for құрылымында көрсетілсе, бұл кедергі жойылады. Fortran үшін nowait тармағы OpenMP do құрылымында емес, оның орнына END DO құрылымында пайда болады. Nowait бір параллель аймақта бірнеше тәуелсіз ілмектер болса өте пайдалы. Бір циклде жұмысын аяқтаған ағындар барлық ағындардың алдыңғы циклде жұмысын аяқтауын күтпестен келесі циклге өтуі мүмкін.

Бөлімдер(Sections) конструкциясы.

Кейде ағындар арасында бөлінетін жұмыс итеративті емес. Бұл жағдайда OpenMP ‘sections’ директивасы OpenMP конструкциясын пайдаланып бөлек ағындардағы код блоктарын орындауға мүмкіндік береді. Бұл жағдайда әрбір бөлім тек бір рет орындалады және бір ағын тек бір бөлімді орындайды.

C/C++ үшін	***Fortran үшін***
#pragma omp sections [опция[[,] опция]. . .] { [#pragma omp section] …… code block 1 ..... [#pragma omp section	!$omp sections [опция[[,] опция]. . .] [!$omp section] …… code block 1 ..... [!$omp section …… code block 2 .....]

…… code block 2 .....] . . . }	. . . !$omp end sections

Жеке(Single) конструкциясыда параллель аймақта пайда болуы мүмкін. Single директивасы келесі код блогы параллельді топта тек бір ағынмен орындалуы керек екенін көрсетеді. Қандай ағын қолданылатыны анық емес.

C/C++ үшін	***Fortran үшін***

#pragma omp single [опция[[,] опция]. . .]
{
…… код .....
}
	!$omp single [опция[[,] опция]. . .]
	…… код.....
	!$omp end single

Мастер конструкциясы.

Мастер конструкциясы тек негізгі ағынмен орындалуына кепілдік берілген код блогын анықтайды. Оның кіруге немесе шығуға ешқандай кедергісі жоқ. Кедергі талап етілмейтін жағдайларда, жеке конструкциямен салыстырғанда мастер конструкциясы артық болуы мүмкін. Мастер конструкциясы деректерді инициализациялау үшін жиі пайдаланылады (кедергі конструкциясымен бірге).

C/C++ үшін	***Fortran үшін***
#pragma omp master …… код	!$omp master …… код..... !$omp end master

Деректер жарысы (Data Race condition)

Деректер жарысы көп ағындар бірдей ортақ деректерді бір уақытта оқығанда немесе жазғанда пайда болады.

Мысал: екі ағынның әрқайсысы ортақ бүтін айнымалының мәнін бірге арттырады (23-сурет).

Correct sequence

Thread 1	Thread 2		value
			0
read value		←	0
Increase value			0
write back		→	1
	read value	←	1
	increase value		1
	write back	→	2

Incorrect sequence

Thread 1	Thread 2		value
			0
read value		←	0
	read value	←	0
increase value			0
	increase value		0
write back		→	1
	write back	→	1

23 - сурет. Бүтін айнымалының мәнін бірге арттыратын екі ағынды алгоритмнің дұрыс және қате реттілігі

Бұндай жағдайларды болдырмау үшін *ағындарды синхрондау* әдісін қолдануға болады. Синхрондаудың 3 түрі бар:

- Critical - директива бір уақытта тек бір ағынға берілген кодқа қол жеткізуге мүмкіндік береді.
- Barrier - топтағы барлық ағындарды синхрондайды. Әрбір кедергіге топтағы барлық ағындар кездесуі керек немесе мүлде болмауы керек. Жұмысты бөлісу аймақтарының және кездесетін кедергі аймақтарының реті топтағы әрбір ағын үшін бірдей болуы керек.
- Atomic - атомдық конструкция бірнеше ағындарға ортақ айнымалы мәнді қауіпсіз жаңартуға мүмкіндік береді. Деректер жарысы мәселесін шешуде бірінші атомдық конструкцияны пайдаланған жөн (дұрыс, бірақ баяу).

Мысал: Локальді қосындыларды параллель есептеу (24 - сурет). Мұндағы m – ағындар саны, n – матрица ұзындағы, LS – локальді қосынды.

Thread 1	Thread 2	.....	Thread m
Read initial S			
$S = S + LS_1$			
Write S			
	Read S		
	$S = S + LS_2$		
	Write S		
		.....	
			Read S
			$S = S + LS_m$
			Write S

24 - сурет. Atomic синхрондау түріне мысал

Тапсырма:

OpenMP технологиясын пайдаланып, векторды екіөлшемді матрицаға көбейтетін C тілінде жазылған бағдарламаны Param–Bilim суперкомпьютерінде орындаңыз.

Бақылау сұрақтары:

1. Бағдарламалау тіліне арналған OpenMP кеңейтімдерін атаңыз.
2. OpenMP конструкцияларын атаңыз.
3. Параллель аймақ конструкциясы қандай функция атқарады?
4. Опция дегеніміз не? OpenMP параллель аймығы үшін қол жетімді қандай опцияларды білесіз?
5. Жұмысты бірлесіп орындау кеңейтімінің конструкцияларын атаңыз.