**Упражнение 8**

OpenMP Клауза *SCHEDULE*

Клауза *SCHEDULE* се използва с *for.* Има следния синтaксис:

**schedule(type[,size/chunk])**

Параметри

**Type**

Видът на насрочването(scheduling), може да бъде dynamic, guided, runtime, или static.

**Size(Chunk)**

(Незадължителен) Определя размера на повторенията. Трябва да е цяло число. Не важи, когато типът е runtime.

Клауза *SCHEDULE* е полезна, ако конструкцията за споделяне на работата е do-loop или for-loop. Итераците в конструкцията за споделяне на работа са присвоени на нишки в съответствие с метода на насрочване(scheduling), определен от тази клауза.

**static**: Тук итерациите са разпеделени между всички нишки още преди да се изпълнят от цикъла. Итерациите по подразбиране са разделени между нишките. Определянето на цяло число за chunk параметъра ще разпредели брой съседни итерации на една нишка.

**dynamic**: Тук някои от итерациите са разпределени на по-малък брой нишки. След като определена нишка завърши разпределената си итерация, тя се връща, за да получи още една от итерациите, които са останали. Пакетът от параметри определя броя на непрекъснатите повторения, които са разпределени в дадена нишка в даден момент.

**guided**: Голяма част от съседни итерации се разпределя динамично за всяка нишка. Размерът на chunk-a намалява експоненциално с всяко последователно разпределение до минимален размер, определен в параметърa chunk.

**Прегледайте / Компилираите / Изпълнете примерния код**

Този пример демонстрира споделяне на работата на цикъл for в OpenMP. Забележете, че се определя динамичното насрочване(scheduling) на нишките и се назначава определен брой итерации, които трябва да бъдат направени от всяка нишка.

Задайте броя на нишките на 4.

Прегледайте резултата. Вижте как всъщност итерациите на цикъла са насрочени(scheduled) в групата от нишки.

Пуснете програмата още няколко пъти и прегледайте резултата. Обикновено динамичното насрочване(scheduling) не е детерминирано. Всеки път, когато стартирате програмата, различни нишки могат да изпълняват различни части от работата. Възможно е дори една нишка да не свърши работа, защото друга нишка е по-бърза и отнема повече работа. Всъщност даже е възможно една нишка да свърши цялата работа.

Редактирайте кода и променете динамичното планиране на статично.

Прекомпилирайте и стартирайте модифицираната програма. Забележете разликата в производителността в сравнение с динамичното планиране. По-точно, забележете, че нишката 0 получава първата част, нишката 1 - втората част и т.н.

Пуснете програмата още няколко пъти. Променя ли се резултата? При статично планиране разпределението на работата е детерминирано и не трябва да се променя между стартиранията и всяка нишка получава работа.

Помислете върху възможните разлики в производителността между динамично и статично планиране.

