Chapter 23 Affinity

1. OpenMP thread affinity control

Thread placement can be controlled with two environment variables:

OMP_PROC_BIND —> how threads are bound to OpenMP places;

OMP_PLACES —> describes these places in terms of the available hardware.

OMP_PLACES = sockets, cores, threads

setting OMP_PLACES = threads ties each OpenMP thread to a specific hardware thread.

Place definition, location: number: stride

OMP_PLACES = “{0:8:1}, {8:8:1}”

OMP_PROC_BIND = close, spread, true, false, master

false: set no binding

true: lock threads to a core

master: collocate threads with the master thread

close: place threads close to the master in the places list

spread: spread out threads as much as possible

2. First-touch

Memory allocated with malloc and like routines is not actually allocated, that only happens when data is written to it. The first thread that writes data is its memory cache that that data is stored to.

3. Affinity control outside OpenMP

Process placement can be controlled on the Operating system level by *numactl* on Linux; *start/affinity* on Windows.

Chapter 24 Memory Model

1. Thread synchronization

The producer-consumer model, where one section, the producer, sets a flag when data is available, and the other, the consumer, waits until the flag is set.

2. Data races

OpenMP, any memory location that is written by one thread can not be read by another thread in the same parallel region if no synchronization is done.

Without synchronization, threads are allowed to maintain a value for a variable locally that is not the same as the stored value.

3. Relaxed memory model

flush

there is an implicit flush of all variables at the start and end of a parallel region.

there is a flush at each barrier, whether explicit or implicit, such as at the end of a work sharing.

at entry and exit of a critical section

when a lock is set or unset

Questions:

1. What does OMP_PLACES = “{0:4:8} : 4 : 1” mean?

2. How to explain the results on page 277?