Nome e Cognome _____ Matricola _____

Corso di *High Performance Computing*
Ingegneria e Scienze Informatiche—Università di Bologna

Prova Scritta, 22/1/2019

- La prova dura 60 minuti
- Durante la prova non è consentito consultare libri, appunti o altro materiale.
- Non è consentito l'uso di dispositivi elettronici (ad esempio, cellulari, tablet...), né interagire in alcun modo con gli altri studenti pena l'esclusione dalla prova, che potrà avvenire anche dopo il termine della stessa.
- Le risposte devono essere scritte **a penna** su questi fogli, in modo **leggibile**. Le parti illeggibili o scritte a matita saranno ignorate.
- Eventuali altri fogli possono essere utilizzati per la brutta copia ma non verranno valutati.
- I voti saranno pubblicati su AlmaEsami e ne verrà data comunicazione all'indirizzo mail di Ateneo (@studio.unibo.it).
- I voti restano validi fino alla sessione d'esame di **settembre 2019** inclusa. Dopo tale data tutti i voti in sospeso saranno persi.

NON SCRIVERE NELLA TABELLA SOTTOSTANTE

D. 1	D. 2	D. 3	D. 4
/ 8	/ 8	/ 8	/ 8

Domanda 1.

a) Descrivere brevemente le caratteristiche principali delle architetture SIMD.
b) Fornire un esempio di applicazione che si ritiene adatta ad essere realizzata mediante il paradigma SIMD, motivando la risposta.
c) Fornire un esempio di applicazione che si ritiene NON adatta ad essere realizzata mediante il paradigma SIMD, motivando la risposta.

Commento: Relativamente al punto c) non è vero che le computazioni di tipo stencil sono inadatte al paradigma SIMD; quindi ad esempio il Game of Life non è un buon esempio di applicazione NON adatta ad essere realizzata mediante il paradigma SIMD, perché si può realizzare la condizione necessaria a determinare lo stato successivo usando "selection and masking". Altre computazioni di tipo stencil non necessitano di condizioni e quindi si prestano immediatamente ad una implementazione SIMD.

Domanda 2. Si consideri il seguente frammento di codice in linguaggio C:

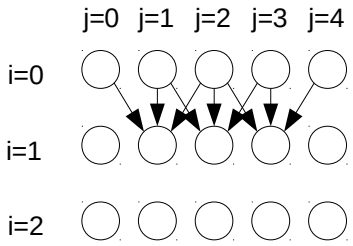
```
#define N …
double A[N][N];

/* … */

for (int k=0; k<500; k++) {
   for (int i=1; i<N; i++) {
      for (int j=1; j<N-1; j++) {
         A[i][j] = A[i][j] + (A[i-1][j-1] + A[i-1][j] + A[i-1][j+1])/3.0;
      }
   }
}
```

Si spieghi in quale posizione è possibile inserire un costrutto `#pragma omp parallel for` per ottenere una versione parallela del programma che risulti equivalente a quella seriale. Il codice non deve essere modificato. Giustificare la risposta.

Commento: Il ciclo parallelizzabile è quello più interno (`for (int j=1; j<N-1; j++) { … }`). In esercizi come questo è sempre una buona idea mostrare il diagramma delle dipendenze; limitatamente ai cicli più interno, si ottiene quanto segue:



Da questo si può osservare come sia possibile calcolare in parallelo i valori della stessa riga, e quindi si può parallelizzare il ciclo più interno. È sbagliato parallelizzare gli altri due; in particolare, le iterazioni del ciclo più esterno non sono indipendenti, dato che i valori $A[i][j]$ calcolati al termine di una iterazione su k dipendono dai valori calcolati al termine dell'iterazione precedente;

Domanda 3.

a) Che cosa si intende con *sbilanciamento del carico*?
b) Descrivere (a parole) una applicazione in cui può verificarsi sbilanciamento del carico, motivando la risposta (è possibile fare riferimento a quanto visto a lezione e/o in laboratorio).
c) Spiegare in quali modi, in generale, è possibile porre rimedio allo sbilanciamento del carico.

Commento: Il punto chiave nella risposta a) è che lo sbilanciamento del carico ha a che fare con una quantità di "lavoro" molto differente tra i vari processi/thread, non necessariamente "quantità di dati". Infatti, nel calcolo dell'insieme di Mandelbrot (citato da molti, correttamente, come esempio in cui si può verificare sbilanciamento del carico) lo sbilanciamento del carico si può verificare anche nel caso in cui si partizioni il dominio in modo perfettamente bilanciato: in tal caso i processi elaborano lo stesso numero di pixel, ma si verifica comunque sbilanciamento del carico perché il tempo necessario per determinare il colore di ciascun pixel non è uniforme.
Nel punto c) è importante segnalare l'uso del paradigma master-worker combinato con un partizionamento a grana più fine (tenuto conto che una grana troppo fine potrebbe produrre overhead, come spiegato a lezione).

Domanda 4. Descrivere una primitiva di comunicazione collettiva MPI, a scelta. Specificarne il comportamento (non è necessario descriverne la sintassi), e illustrare un caso d'uso concreto in cui può essere utilizzata.

Commento. In alcuni casi la spiegazione della primitiva è stata poco precisa; in questa domanda sarebbe stato utile combinare una spiegazione “a parole” con un diagramma esplicativo.