Programación Paralela

Exámen Parcial
2020-II
Prof. José Fiestas

Indicaciones específicas:
Duración: 120 minutos
Número de preguntas: 5
Solo está permitido el uso de material de clase durante el exámen. Ningún otro material.
La entrega se hará por Classroom (pdf o foto)
No necesita enviar códigos solución a los ejercicios, pero no está prohibido

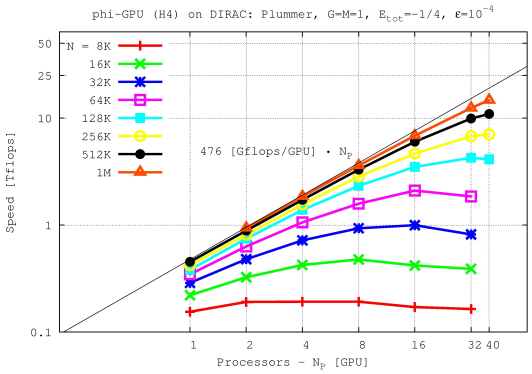
---

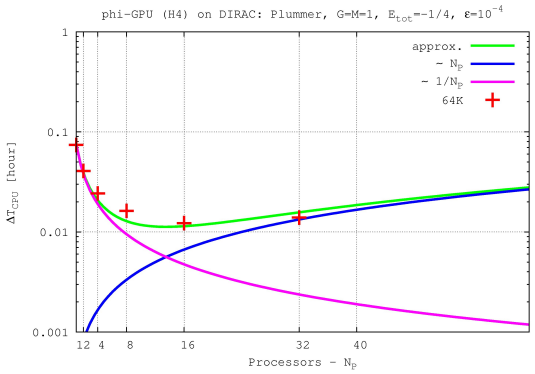
La rúbrica en cada pregunta tiene dos puntos escenciales:
- Desarrollo analítico o numérico de los ejercicios (70%)
- Argumentación de resultados obtenidos (30%)

Pregunta1 $^{(a1)}$

1a)

Las siguientes gráficas representan velocidad en FLOPs vs. número de nodos de un código de N-cuerpos para N=64K, así como el tiempo de ejecución en horas vs. número de nodos. Describa cómo ambos gráficos describen el mismo efecto en performance ¿ Cúal es el número óptimo de nodos para éste modelo? **(2 puntos)**





Programación Paralela

Exámen Parcial
2020-II
Prof. José Fiestas

1b)
Un algoritmo paralelo de ordenamiento por comparación tiene una complejidad

$$T(p) = O(\frac{nlog^2n}{p})$$

Determine la velocidad y eficiencia absolutas (i.e. considere el algoritmo de ordenamiento secuencial por comparación de mejor complejidad)
¿Cómo depende el número de procesos p de la cantidad de elementos n, si la velocidad tiene una complejidad constante?
¿Considera que éste último caso representa un algoritmo escalable?
(2 puntos)

Pregunta(2)$^{(a1)}$ (4 puntos)
Defina la función suma, a ejecutarse en paralelo bajo las siguientes condiciones:
- sumar el vector local A (de dimensión n) con la parte correspondiente del vector B
- copiar el resultado en un vector local C.

```
void suma(double Aloc[], double B[], double C[], int n, int size, int rank)
```

Donde rank es el nombre del proceso, de un total size de procesos
Argumente su implementación

Pregunta(3)$^{(b1)}$ (4 puntos)
¿Qué ejecuta el siguiente código? ¿Considera que está correctamente paralelizado? Argumente sus respuestas, y en caso encuentre un error, proponga una solución.

```
double A[N], B[N];
int  rank, size, origen, destino;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &size);
origen = (rank==0)? size-1: rank-1;
destino= (rank==size-1)? 0: rank+1;
MPI_Ssend(A, N, MPI_DOUBLE, destino, 1, MPI_COMM_WORLD);
MPI_Recv(B, N, MPI_DOUBLE, origen, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
```

Considere que el fragmento de código es parte de un código correctamente programado. Observe el uso de Ssend (Send sincrónico)

Programación Paralela
Exámen Parcial
2020-II
Prof. José Fiestas

Pregunta(4)$^{(b1)}$
Se quiere paralelizar el siguiente código. Se dispone de 3 procesos.

```
double a[N],b[N],c[N],v=0.0,w=0.0;
T1(a,&v);
T2(b,&w);
T3(b,&v);
T4(c,&w);
T5(c,&v);
T6(a,&w);
```

Todas las funciones leen y modifican ambos argumentos, también los arrays. Suponemos que los vectores a, b y c están almacenados en P0, P1 y P2, respectivamente.
a) Dibuje el DAG de las diferentes tareas, indicando qué tarea se asigna a cada proceso (2 puntos)
b) Describa la estructura principal de un código MPI que resuelva el problema. (3 puntos)

Pregunta(5)$^{(a1)}$ (3 puntos)
Encuentre por lo menos 3 errores en el siguiente código en MPI

```
[...]
long localmax;

if (rank != 1) {
  for (int i = 0; i < num_workers-1; i=i+2) {
      MPI_Irecv(&localmax, 1, MPI_LONG, i,  123, MPI_COMM_WORLD, &request);
  }
  MPI_Wait(&request, &status);
  globalmax = findMaxValue(localmax, globalmax);
  doImportantCalculation(globalmax);

} else {
  MPI_Isend(&localmax, 1, MPI_LONG, 1, 123, MPI_COMM_WORLD, &request);
  someFooBarLongCalc();
  MPI_Wait(&request, &status);
}
[...]
```

Programación Paralela
Exámen Parcial
2020-II
Prof. José Fiestas

$^{(a1)}$ **outcome a1: Aplicar conocimientos de computación y de matemáticas apropiadas para la disciplina**
$^{(b1)}$ **outcome b1 : Analizar problemas e identificar y definir los requerimientos computacionales apropiados para su solución.**