

Esquemas Algorítmicos Paralelos

Departamento de Electrónica y Sistemas

Primavera 2016



1 Single Process Multiple Data



Single Process Multiple Data

- Este tipo de algoritmos surgen de un particionado de dominio sobre grandes arrays de datos.
- Se divide el espacio de datos en regiones y se asignan a procesadores intentando minimizar el coste de comunicaciones y maximizar la eficiencia de cada nodo.
- Para ser rentable, el volumen de computación debe ser al menos un orden mayor que el de comunicaciones.
- Ejemplo: multiplicación de matrices.
 - Computación: $\Theta(n^3)$
 - Datos: $\Theta(n^2)$

Multiplicación de Matrices

Descomposición y asignación

- La descomposición de dominio es la que mejor se adapta a este problema.
- Dado que el balanceo de carga no es un problema, se selecciona una división en bloques de filas dependiente del número de procesos.
- La matriz B se replica en todos los procesos. Así, cada uno es capaz de obtener las mismas filas en x que posee en A sin comunicaciones.
- Inicialmente, es necesario hacer el reparto usando operaciones *scatter* por parte del proceso encargado de la E/S.
- La matriz x global se ensambla usando un *gather*.

Multiplicación de matrices

Reparto de datos

```
// Entrada: matrices A y B, dimensiones n x n
// Sólo disponibles en el proceso 0
// Suponemos que todos los procesos disponen de
// MPI_COMM_WORLD, my_id, numprocs
// Por simplicidad se omite la declaración de
// variables

// Distribución de n y la matriz B
MPI_Bcast( &n, 1, MPI_INT, 0, MPI_COMM_WORLD );
MPI_Bcast( B, n*n, MPI_DOUBLE, 0, MPI_COMM_WORLD );

// Cálculo del número de filas por proceso
rows = ceil( n / numprocs );

// Corrección del tamaño de A para scatter
if( (my_id == 0) && (n% numprocs) ) {
    A = (double *)realloc( A,
        sizeof(double)*n*numprocs*rows );
}

// Reserva de memoria para las submatrices local
// Nota: el proceso (p-1) puede sobrerreservar
Alocal = (double*)malloc( sizeof(double)*rows*n );
xlocal = (double*)malloc( sizeof(double)*rows*n );
```

Multiplicación de matrices

Reparto de datos

```
// Scatter de los datos de A
MPI_Scatter( A, rows*n, MPI_DOUBLE, Alocal, rows*n,
            MPI_DOUBLE, 0, MPI_COMM_WORLD );

// Corrección del número de filas en (p-1)
if( my_id == numprocs-1 ) {
    rows = n - rows*(numprocs-1);
}

// Lazo computacional
for( i = 0; i < rows; ++i ) {
    for( j = 0; j < n; ++j ) {
        xlocal[i][j] = 0;
        for( k = 0; k < n; ++k ) {
            xlocal[i][j] += Alocal[i][k] * B[k][j];
        }
    }
}
```

Multiplicación de matrices

Reparto de datos

```
// (Sobre)reserva para el vector x en el proceso 0
if( my_id == 0 ) {
    x = (double*)malloc(
        sizeof(double)*n*numprocs*rows );
}

// Gather de todos los datos
MPI_Gather( xlocal, n*rows, MPI_DOUBLE, x, n*rows,
    MPI_DOUBLE, 0, MPI_COMM_WORLD );
```

Variantes

- Distribución bidimensional.
- Distribución de ambas matrices.
- Tiling (blocking).