PRODUCTO MATRIZ-VECTOR CON MATRIZ BANDA

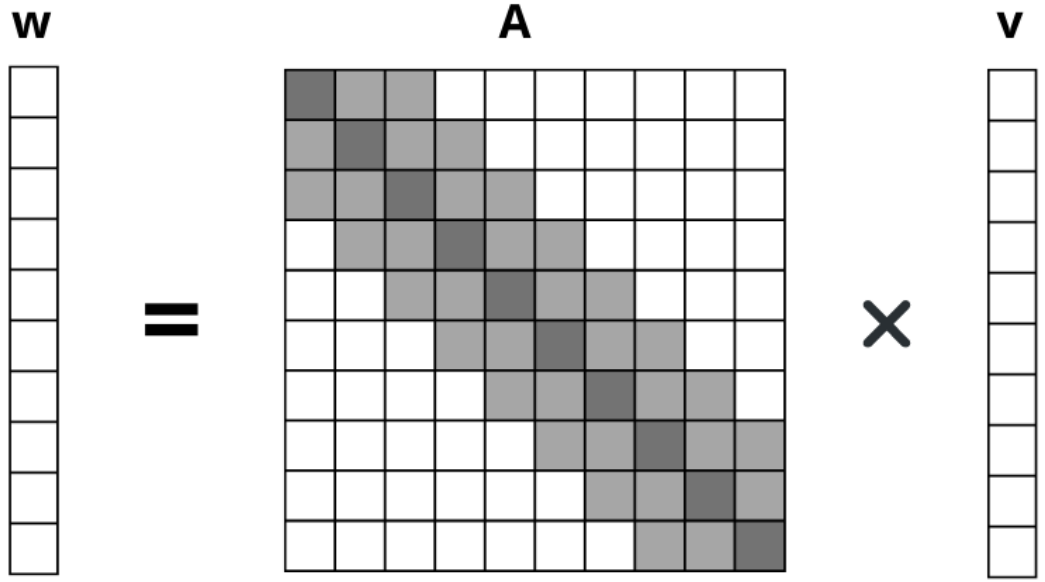
JAVIER MELIÁ SEVILLA

GUILLERMO ATO MORCILLO

DESCRIPCIÓN DEL PROBLEMA

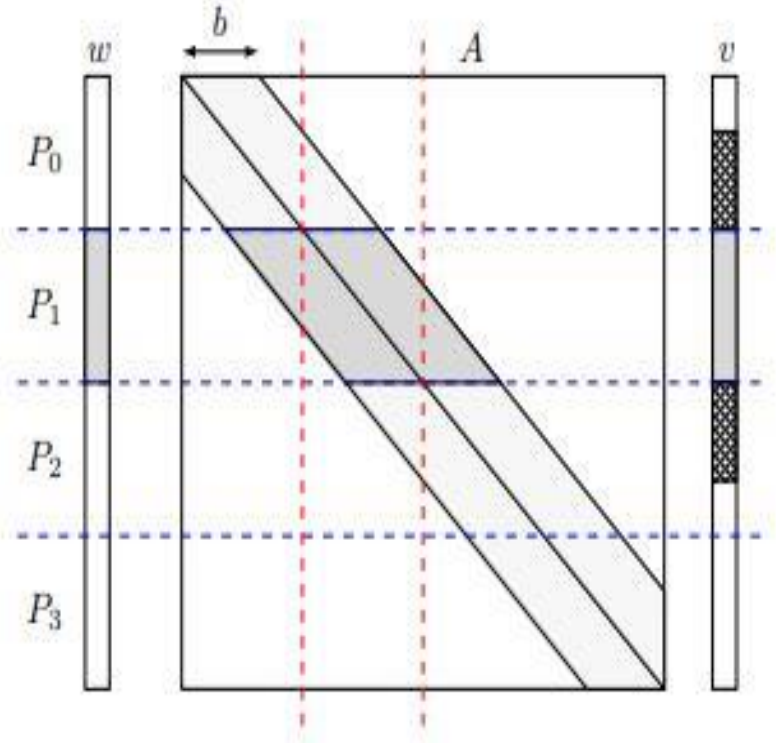
PRODUCTO MATRIZ-VECTOR CON MATRIZ BANDA

- Producto matriz-vector $w = Av$
- Estructura banda: a_{ij} es cero si $|i - j| > b$



DESCRIPCIÓN DE LA PARALELIZACIÓN REALIZADA

VERSIÓN PARALELA BÁSICA

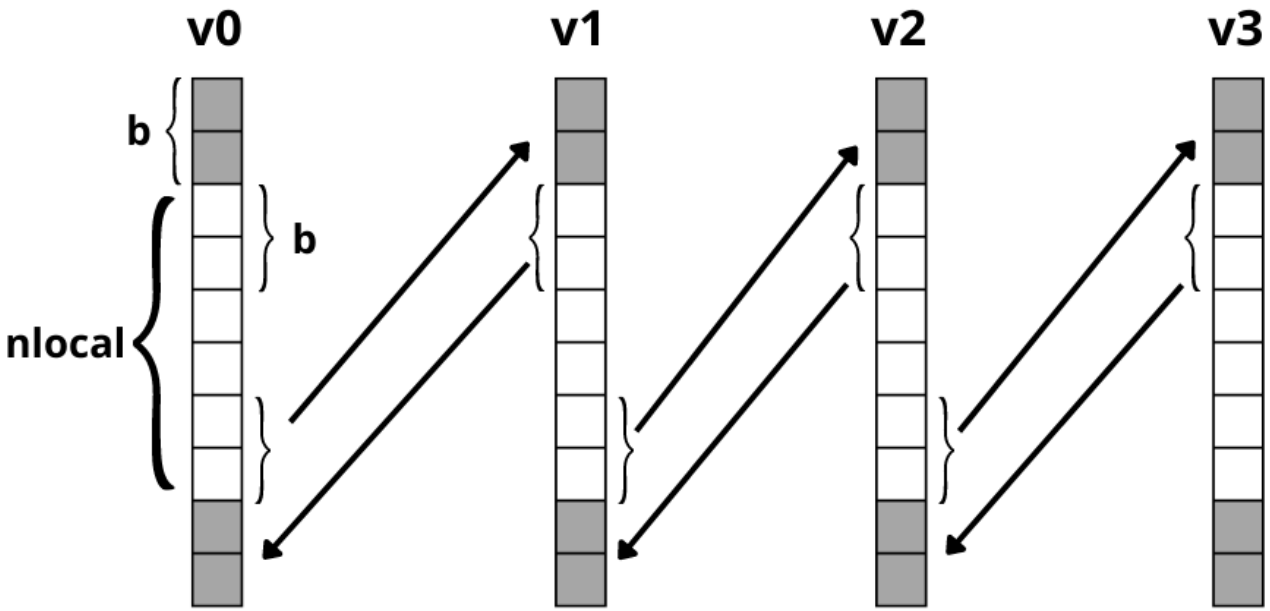


1. A = nLocal * N
2. w = nlocal
3. v = nlocal + 2b
4. Con MPI_Sendrecv
5. MPI_Gather para recibir W en el proceso 0

INTERACCIÓN ENTRE PROCESOS

Vector v

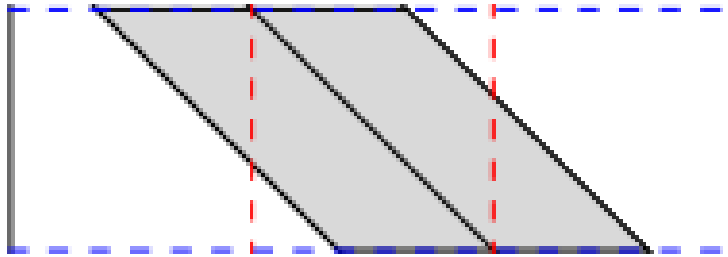
- b oscuro = memoria a recibir
- b blanco = memoria a enviar
- Enviar b elementos al vecino ↑
- Enviar b elementos al vecino ↓

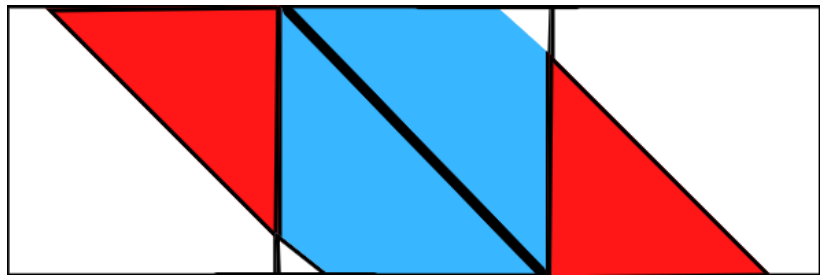


PARALELIZACIÓN CON SOLAPAMIENTO

Secuencia

- Enviar b elementos respectivos a cada proceso
- Computar la zona intermedia (azul)
- Sincronizar procesos
- Computar el resto (rojo)





VERSIÓN SECUENCIAL VS PARALELA BÁSICA

RESERVA DE MEMORIA

VERSIÓN SECUENCIAL

```
/* Reserva de memoria */
A = (double*)calloc(N*N,sizeof(double));
v = (double*)calloc(N,sizeof(double));
w = (double*)calloc(N,sizeof(double));
```

VERSIÓN PARALELA BÁSICA

```
nLocal= N/size;
t1 = MPI_Wtime();

/* Reserva de memoria */
A = (double*)calloc(nLocal*N,sizeof(double));
v = (double*)calloc(nLocal+(2*b),sizeof(double));
w = (double*)calloc( nLocal,sizeof(double));
aux = (double*)calloc( N,sizeof(double));
```

INICIALIZACIÓN DE DATOS

SECUENCIAL

```
/* Inicializar datos */
for (i=0; i<N; i++) A[i*N+i] = 2*b;
for (i=0; i<N; i++) {
  for (j=0; j<N; j++) {
    if (i!=j && abs(i-j)<=b) A[i*N+j] = -1.0;
  }
}
for (i=0; i<N; i++) v[i] = 1.0;
```

PARALELA BÁSICA

```
/* Inicializar datos */
for (i=0; i<nLocal; i++){
  iglobal = (rank*nLocal)+i;

  A[i*N+iglobal] = 2*b;
}
for (i=0; i<nLocal; i++) {
  iglobal = (rank*nLocal)+i;
  for (j=0; j<N; j++) {
    if (iglobal!=j && abs(iglobal-j)<=b) A[i*N+j] = -1.0;
  }
}

for (i=b; i<nLocal+b; i++) v[i] = 1.0;
```

CALCULO DEL VECTOR W

```
void matvec(int N,int b,double *A, double *v, double *w)
{
  int i, j, li, ls;

  for (i=0; i<N; i++) {
    w[i] = 0.0;
    li = i-b<0? 0: i-b;  /* limite inferior */
    ls = i+b>N-1? N-1: i+b;  /* limite superior */
    for (j=li; j<=ls; j++) {
      w[i] += A[i*N+j]*v[j];
    }
  }
}
```

CALCULO DEL VECTOR W

```
void matvec_parallel(int N,int nLocal, int b,double *A, double *v, double *w)
{
  int i, j, li, ls,size,rank,next,prev,jglobal;

  MPI_Comm_size(MPI_COMM_WORLD,&size);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Status status;
  if (rank == 0) prev = MPI_PROC_NULL;
  else prev = rank-1;
  if (rank == size-1) next = MPI_PROC_NULL;
  else next = rank+1;
  MPI_Sendrecv(&v[b], b, MPI_DOUBLE, prev, 0, &v[nLocal+b], b, MPI_DOUBLE, next, MPI_ANY_TAG, MPI_COMM_WORLD, &status);
  MPI_Sendrecv(&v[nLocal], b, MPI_DOUBLE, next, 0, &v[0], b, MPI_DOUBLE, prev, MPI_ANY_TAG, MPI_COMM_WORLD, &status);

  for (i=0; i<=nLocal-1; i++) {
    w[i] = 0.0;
    jglobal = (rank*nLocal)+i;
    li = jglobal-b<0? 0: jglobal-b;  /* limite inferior */
    ls = jglobal+b>N-1? N-1: jglobal+b;  /* limite superior */
    for (j=li; j<=ls; j++) {
      w[i] += A[i*N+j]*v[j-(rank*nLocal)+b];
    }
  }

}
```

MOSTRAR SOLUCIÓN

SECUENCIAL

```
/* Imprimir solución */
if (N<100) for (i=0; i<N; i++) printf("w[%d] = %g\n", i, w[i]);
printf("El tiempo es %g\n", t2-t1);
```

PARALELA BÁSICA

```
/* Imprimir solución */
MPI_Gather(&w[0], nLocal, MPI_DOUBLE, aux, nLocal,MPI_DOUBLE,0,MPI_COMM_WORLD);
t2 = MPI_Wtime();

 if(rank==0){
  if (N<100) for (i=0; i<N; i++) printf("w[%d] = %g\n", i, aux[i]);
  printf("El tiempo es %g\n", t2-t1);
 }
```

VERSIÓN PARALELA CON SOLAPAMIENTO

CALCULO DEL VECTOR W

```
MPI_Isend(&v[b], b, MPI_DOUBLE, prev, 0, MPI_COMM_WORLD, &request[0]);
MPI_Irecv(&v[nLocal+b], b, MPI_DOUBLE, next, MPI_ANY_TAG,  MPI_COMM_WORLD, &request[1]);
MPI_Isend(&v[nLocal], b, MPI_DOUBLE, next, 0, MPI_COMM_WORLD, &request[2]);
MPI_Irecv(&v[0], b, MPI_DOUBLE, prev, MPI_ANY_TAG, MPI_COMM_WORLD, &request[3]);

for (i=0; i<=nLocal-1; i++) {
  w[i] = 0.0;
  jglobal = (rank*nLocal)+i;
  li = jglobal-b<0? 0: jglobal-b;  /* limite inferior */
  ls = jglobal+b>N-1? N-1: jglobal+b;  /* limite superior */
  for (j=li; j<=ls; j++) {
    if(j >= b && j < b+nLocal){
      w[i] += A[i*N+j]*v[j-(rank*nLocal)+b];
    }
  }

}
MPI_Waitall(4, request, MPI_STATUS_IGNORE);
for (i=0; i<=nLocal-1; i++) {
  jglobal = (rank*nLocal)+i;
  li = jglobal-b<0? 0: jglobal-b;  /* limite inferior */
  ls = jglobal+b>N-1? N-1: jglobal+b;  /* limite superior */
  for (j=li; j<=ls; j++) {
    if(j < b || j >= b+nLocal){
      w[i] += A[i*N+j]*v[j-(rank*nLocal)+b];
    }
  }
}
```

VALIDACIÓN

SECUENCIAL VS BÁSICA VS SOLAPAMIENTO

```
jamese@alumno.upv.es@kahan:~/mpi/trabajo$ ./secuencial 30 3
w[0] = 3
w[1] = 2
w[2] = 1
w[3] = 0
w[4] = 0
w[5] = 0
w[6] = 0
w[7] = 0
w[8] = 0
w[9] = 0
w[10] = 0
w[11] = 0
w[12] = 0
w[13] = 0
w[14] = 0
w[15] = 0
w[16] = 0
w[17] = 0
w[18] = 0
w[19] = 0
w[20] = 0
w[21] = 0
w[22] = 0
w[23] = 0
w[24] = 0
w[25] = 0
w[26] = 0
w[27] = 1
w[28] = 2
w[29] = 3
```

```
jamese@alumno.upv.es@kahan:~/mpi/trabajo$ mpiexec -n 3 ./sinsolapamiento 30 3
w[0] = 3
w[1] = 2
w[2] = 1
w[3] = 0
w[4] = 0
w[5] = 0
w[6] = 0
w[7] = 0
w[8] = 0
w[9] = 0
w[10] = 0
w[11] = 0
w[12] = 0
w[13] = 0
w[14] = 0
w[15] = 0
w[16] = 0
w[17] = 0
w[18] = 0
w[19] = 0
w[20] = 0
w[21] = 0
w[22] = 0
w[23] = 0
w[24] = 0
w[25] = 0
w[26] = 0
w[27] = 1
w[28] = 2
w[29] = 3
```

```
jamese@alumno.upv.es@kahan:~/mpi/trabajo$ mpiexec -n 3 ./solapamiento 30 3
w[0] = 3
w[1] = 2
w[2] = 1
w[3] = 0
w[4] = 0
w[5] = 0
w[6] = 0
w[7] = 0
w[8] = 0
w[9] = 0
w[10] = 0
w[11] = 0
w[12] = 0
w[13] = 0
w[14] = 0
w[15] = 0
w[16] = 0
w[17] = 0
w[18] = 0
w[19] = 0
w[20] = 0
w[21] = 0
w[22] = 0
w[23] = 0
w[24] = 0
w[25] = 0
w[26] = 0
w[27] = 1
w[28] = 2
w[29] = 3
```

ESTUDIO DE PRESTACINES

B	C	D	E	F	G	H	I	J
			Nodos = 2		Nodos = 3		Nodos = 4	
Tamaño matriz	Tamaño ancho de banda	Secuencial	Sin solapamiento	Con solpamiento	Sin solapamiento	Con solpamiento	Sin solapamiento	Con solpamiento
40	4	0,38	0,026	0,036	0,02	0,03	0,018	0,02
40	10	0,78	0,04	0,062	0,036	0,06	0,03	0,04
40	20	0,12	0,063	0,092	0,06	0,09	0,045	0,07
100	4	0,09	0,051	0,073	0,04	0,06	0,04	0,05
100	20	0,28	0,164	0,27	0,14	0,22	0,11	0,17
100	50	0,56	0,34	0,5	0,32	0,46	0,22	0,33
200	4	0,14	0,09	0,14	0,07	0,11	0,06	0,09
200	50	1,32	0,75	1,2	0,65	1,02	0,49	0,78
200	100	2,24	1,25	1,93	1,15	1,73	0,84	1,26
500	4	0,37	0,27	0,35	0,38	0,27	0,13	0,21
500	100	6,69	4,47	5,95	3,18	5,04	2,40	3,78
500	250	13,79	10,2	11,85	10,38	10,77	5,24	7,68
1000	4	0,77	0,62	0,695	0,33	0,53	0,26	0,4
1000	250	32,5	29,1	31,01	15,67	24,6	11,76	18,98
1000	500	55,46	47,11	46,9771	36,73	42,56	20,54	30,7

TIEMPOS

SE HA UTILIZADO LA OPCIÓN –O PARA COMPILAR PARA UNA MAS OPTIMIZADA