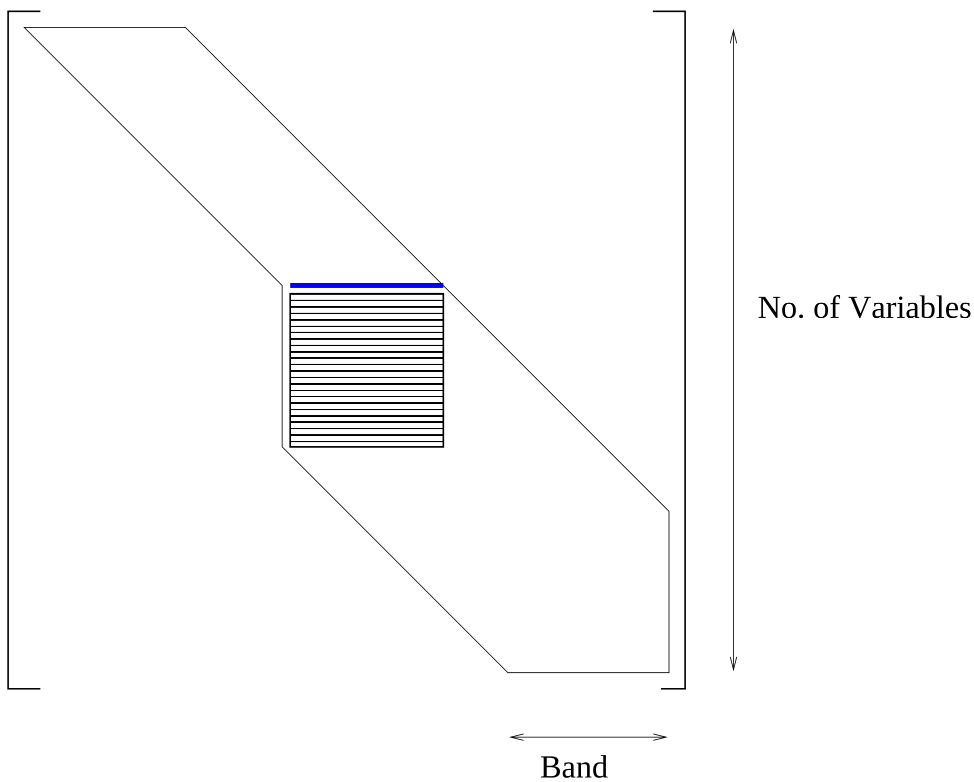
PARALLEL PERFORMANCE



Example: Finite element code to compute liquid crystal flows. Wall clock times on otherwise idle machines.

Sun 2–Processor UltraSparc (50 time steps)

Mesh	Variables	Band	Scalar	2–Processors	Ratio
8×8	1307	173	8:33	5:29	0.64
16×16	4779	309	94:58	49:48	0.52

Dual Processor Pentium 450 (50 time steps)

Mesh	Variables	Band	Scalar	2–Processors	Ratio
8×8	1307	173	5:38	6:06	1.08
16×16	4779	309	89:51	58:47	0.65

UltraSparc OPEN MP CODE

```
#include<stdio.h>

extern int Band;

void solve(int nn,double a[], double f[])
{
 int i,j,k,ii,jj,jkmax;  /* Variables labeled 0,1,2, ... ,nn-1 */
 double temp;            /* a[i,j] = a[i*(2*Band-1) + j-i ]    */

 for(i = 0; i < nn-1; i++) /* Gauss Eliminate */
   {
   ii = i * (2*Band-1);
   jkmax = imin(nn,i+Band);

   if( a[ii] == 0.0 ) goto error;

  #pragma MP taskloop private(jj,k,temp)
  #pragma MP taskloop readonly(i,ii,jkmax,Band)

   for(j = i+1; j < jkmax; j++)
     {
     jj = j * (2*Band-1);

     temp = a[jj + i-j] / a[ii];
     if(temp == 0.0) continue;

     f[j] -= temp * f[i];

     for(k = i+1; k < jkmax; k++) a[jj+k-j] -= temp * a[ii+k-i];
     }

   }
/ * * * * * * * * * Back Substituion Loop Here * * * * * * * * * /
}
```

LINUX P–THREAD CODE

```
#include<stdio.h>
#include<pthread.h>

#define Nproc 2

void * inner(void *);
int imin(int, int);
void solve(int,double *, double *);

volatile extern int Band;

typedef struct
  {
  int i;
  int ii;
  int jp;
  int jkmax;
  int jpstep;
  double *f;
  double *a;
  } Idata;
```

```
void solve(int nn,double a[], double f[])
{
 int j,jp;      /* Split the elimination over Nproc processors */
 double temp;   /* Variables labeled 0,1,2, ... ,nn-1  */
                /* A[i,j] = a[i*(2*Band-1) + j-i ]     */

 volatile Idata tdata[Nproc];
 int i,ii,jkmax,jpstep = Band / Nproc + 1;

 int check = 0;
 void *retval;
 pthread_t threads[Nproc];

 for(i = 0; i < nn-1; i++) /* Gauss Eliminate */
   {
   ii = i * (2*Band-1);
   jkmax = imin(nn,i+Band);

   if( a[ii] == 0.0 ) goto error;

   j = 0;

   for(jp = i+1; jp < jkmax; jp += jpstep)
     {
       tdata[j].i  = i;
       tdata[j].ii = ii;
       tdata[j].jp = jp;
       tdata[j].jkmax = jkmax;
       tdata[j].jpstep = jpstep;
       tdata[j].f = f;
       tdata[j].a = a;

       check |= pthread_create(threads+j, NULL,
                         inner, (void *) &(tdata[j]));
```

```
        j++;
      }

    if(check)
      {
      printf("Can't make threads in solve() -- terminating\n");
      exit(1);
      }

    j = 0;

    for(jp = i+1; jp < jkmax; jp += jpstep)
      check |= pthread_join(threads[j++], &retval);

    if(check)
      {
      printf("pthread_join() failed in solve() -- terminating\n");
      exit(1);
      }
    }
```

/ * * * * * * * * Back Substituion Loop Here * * * * * * * * /

```
}
```

```
void * inner(void *vtd)
{
  int i,ii,j,jj,k,jp;
  double temp,*a,*f;
  Idata *td = (Idata *) vtd;

  i  = td->i;
  ii = td->ii;
  jp = td->jp;
  f  = td->f;
  a  = td->a;

  for(j = jp; j < imin(jp+td->jpstep,td->jkmax); j++)
   {
   jj = j * (2*Band-1);

   temp = a[jj + i-j] / a[ii];
   if(temp == 0.0) continue;

   f[j] -= temp * f[i];

   for(k = i+1; k < td->jkmax; k++) a[jj+k-j] -= temp * a[ii+k-i];
   }

 return(NULL);
}
```