Karlsruhe Institute of Technology

Parallel Programming with MPI and OpenMP

OpenMP-Solution for Gauss-Algorithm

Hartmut Häfner

The Gauss Algorithm A*x = b

```
...
!$OMP PARALLEL
nthreads = omp_get_num_threads()
!print*,' nthreads = ',nthreads
!$OMP END PARALLEL

n = INT(nstart*nthreads**(1./3.))
allocate(A(n,n), b(n), x(n), stat=ierr)
if (ierr /= 0) then
  print*,' Allocation of array failed'
  stop
endif

!$OMP PARALLEL PRIVATE(k,i) SHARED(A)
!$OMP DO SCHEDULE(runtime)
do k=1,n
  do i=1,n
    A(i,k)=n-ABS(i-k)
  enddo
enddo
!$OMP END DO
!$OMP END PARALLEL

do i=1,n
  b(i)=FLOAT(i)
enddo
```

The optimized Gauss Algorithm A*x = b

```
off = nthreads - 1
do j=1,n-1
  r = 1.d0/A(j,j)
  do i=j+1,n
    A(i,j) = A(i,j)*r
  enddo
  if (off > 0) then
    do k=j+1,MIN(j+off,n)
      do i=j+1,n
        A(i,k) = A(i,k) - A(i,j)*A(j,k)
      enddo
    enddo
  Endif
!Update of A(n-j,n-j)
!$OMP PARALLEL PRIVATE(k,i) SHARED(A,j,n)
!$OMP DO SCHEDULE(runtime)
  do k=j+1+off,n
    do i=j+1,n
      A(i,k) = A(i,k) - A(i,j)*A(j,k)
    enddo
  enddo
!$OMP END DO
!$OMP END PARALLEL
  do i=j+1,n
    b(i) = b(i) - A(i,j)*b(j)
  enddo
  off = off - 1
  if (off < 0) off = nthreads - 1
enddo
```

```
!Computation of solution x
x(n) = b(n)/A(n,n)
do j=n,2,-1
!$OMP PARALLEL PRIVATE(i) SHARED(A,x,b,j)
!$OMP DO SCHEDULE(static)
  do i=1,j-1
    b(i) = b(i) - A(i,j)*x(j)
  enddo
!$OMP END DO
!$OMP END PARALLEL
  x(j-1) = b(j-1)/A(j-1,j-1)
enddo
```

Performance of Gaus Algorithm

OMP_SCHEDULE="STATIC,1"
KMP_AFFINITY=verbose,granularity=fine,compact,1,0

bwUniCluster
gauss_par

n=2000, 1 core:
real 1.95s
Mflops 2744

n=2519, 2 cores:
real 2.69s
Mflops 3968

n=3174, 4 cores:
Real 4.24s
Mflops 5036

n=4000, 8 cores:
real 8.44s
Mflops 5055

n=5039, 16 cores:
real 12.18s
Mflops 7005

bwUniCluster
gauss_par_opt

n=2000, 1 core:
Real 2.05s
Mflops 2601

n=2519, 2 cores:
real 2.74s
Mflops 3898

n=3174, 4 cores:
Real 4.25s
Mflops 5018

n=4000, 8 cores:
real 8.48s
Mflops 5034

n=5039, 16 cores:
Real 8.37s
Mflops 10199