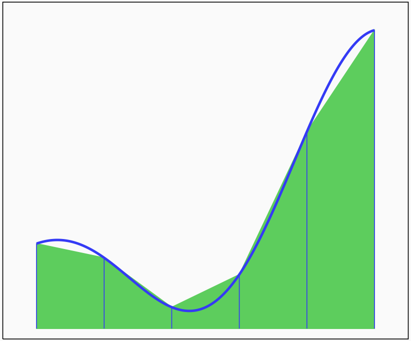
Toward HPC

Chapter 2 – Examples in Shared memory

M1 – MSIAM
February 7, 2018

Example 1: Integral Computation



Trapezium formula is given by $\forall f \in C^2([a;b])$, $\exists \xi \in [a;b]$

$$I = \frac{h}{2}\left(f(a) + 2\sum_{k=1}^{n-1} f(a+kh) + f(b)\right) - (b-a)\frac{h^2}{12}f^{(2)}(\xi)$$

Example 1: Pseudo-code

```
Input b, a, n
h = (b-a)/n
I = (f(a) + f(b))/2.0;
for (i = 0; i<= n-1; i++)
{
  x_i = x_i + h
  I += f(x_i)
}
I = h*I
```

Example 1: Parallel Pseudo-code

```
Find b, a, n
h = (b-a)/n
local_n = n/n_p
local_a = a + id * local_n*h
local_b = local_a + local_n*h
local_I = Trap(local_a, local_b, local_n)

If (id == 0)
{
 I = local_I;
 for (i = 1; i<= n_p; i++)
 {
  I += local_I;
 }
 Display I;
}
```

Example 1: Parallel Code

```
/* Compute the size of intervals */
d = 1.0/(double) n;

/* Start the threads */
#pragma omp parallel default(shared) private(nthreads, tid, x)
{
 /* Get the thread number */
 tid = omp_get_thread_num();

 /* The master thread checks how many there are */
#pragma omp master
 {
  nthreads = omp_get_num_threads();
  printf("Number of threads = %d\n", nthreads);
 }

 /* This loop is executed in parallel by the threads */
#pragma omp for reduction(+:sum)
 for (i=0; i<n; i++) {
  x = d*(double)i;
  sum += f(x) + f(x+d);
 }
} /* The parallel section ends here */

pi = d*sum*0.5;
```

Example 2: Matrix Multiply

Let $A \in \boldsymbol{R}^{n \times m}$ and $B \in \boldsymbol{R}^{m \times p}$. Then $C = A \times B$, $C \in \boldsymbol{R}^{m \times p}$ is defined by

$$\forall 1 \leq i \leq n, 1 \leq j \leq p, c_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj}$$

Example 2: Pseudo-code

```
Input A, B, n, m, p
for(i = 1; i<= n;i++)
{
 for(j = 1; j<= p;j++)
 {
  for(k = 1; k<= m;k++)
  {
   C[i][j]= C[i][j]+ A[i][k] x B[k][j]
  }
 }
}
```

Example 2: Parallel Pseudo-code

```
Input A, B, n, m, p
l_n = n/nn_p
for(i = id*l_n +1; i<= (id+1)*l_n;i++)
{
 for(j = 1; j<= p;j++)
 {
  for(k = 1; k<= m;k++)
  {
   C[i][j]= C[i][j]+ A[i][k] x B[k][j]
  }
 }
}
```

Example 2: Parallel Code

```
#pragma omp parallel shared(a,b,c,nthreads,chunk) private(tid,i,j,k)
{
 tid = omp_get_thread_num();
 /* Initialize matrices */
#pragma omp for schedule (static, chunk)
 for (i=0; i<NRA; i++)
  for (j=0; j<NCA; j++)
   a[i][j]= i+j;
#pragma omp for schedule (static, chunk)
 for (i=0; i<NCA; i++)
  for (j=0; j<NCB; j++)
   b[i][j]= i*j;
#pragma omp for schedule (static, chunk)
 for (i=0; i<NRA; i++)
  for (j=0; j<NCB; j++)
   c[i][j]= 0;

 /* Do matrix multiply sharing iterations on outer loop */
#pragma omp for schedule (static, chunk)
 for (i=0; i<NRA; i++)
  for(j=0; j<NCB; j++)
   for (k=0; k<NCA; k++)
    c[i][j] += a[i][k] * b[k][j];
}
```

Example 3: Gaussian Elimination – Pseudo-code

```
for k = 1 ... m:
 Find pivot for column k:
 i_max  := argmax (i = k ... m, abs(A[i, k]))
 if A[i_max, k] = 0
  error "Matrix is singular!"
 swap rows(k, i_max)
 Do for all rows below pivot:
 for i = k + 1 ... m:
  Do for all remaining elements in current row:
  for j = k ... n:
   A[i, j]  := A[i, j] - A[k, j] * (A[i, k] / A[k, k])
  Fill lower triangular matrix with zeros:
  A[i, k]  := 0
```

Example 3: Parallel Code

```
for(pivot = 1; pivot < n; pivot++)
{
#pragma omp parallel for private(xmult) schedule(runtime)
 {
  for(i = pivot + 1; i < n; i++)
  {
   xmult = a[i][pivot] / a[pivot][pivot];
   for(j = pivot + 1; j < n; j++)
   {
    a[i][j] -= (xmult * a[pivot][j]);
   }
   b[i] -= (xmult * b[pivot]);
  }
 }
}
```

Example 4: Relaxation

- The Gauss-Seidel method is an iterative method to solve linear system of finite dimension $Ax = b$, where A as non-zero diagonal elements, $A \in \boldsymbol{R}^{n\times n}$ and $b \in \boldsymbol{R}^n$.
- The method writes

$$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i}^{n} a_{ij} x_j^{k} \right)$$

Example 4: Sequential code

```c
double *dx = (double*) calloc(n,sizeof(double));
int i,j,k;
for(k=0; k<maxit; k++) {
  double sum = 0.0;
  for(i=0; i<n; i++) {
    dx[i] = b[i];
    for(j=0; j<n; j++){
      dx[i] -= A[i][j]*x[j];
    }
    dx[i] /= A[i][i]; x[i] += dx[i];
    sum += ( (dx[i] >= 0.0) ? dx[i] : -dx[i]);
  }
  printf("%4d : %.3e\n",k,sum);
  if(sum <= epsilon) break;
}
*numit = k+1; free(dx);
```

Example 4: Parallel Code

```
double *dx;
dx = (double*) calloc(n,sizeof(double));
int i,j,k,id,jstart,jstop;
int dnp = n/p;
double dxi;
for(k=0; k<maxit; k++) {
 double sum = 0.0;
 for(i=0; i<n; i++) {
  dx[i] = b[i];
#pragma omp parallel shared(A,x) private(id,j,jstart,jstop,dxi)
  {
   id = omp_get_thread_num();
   jstart = id*dnp;
   jstop = jstart + dnp;
   dxi = 0.0;
   for(j=jstart; j<jstop; j++) {
    dxi += A[i][j]*x[j];
   }
#pragma omp critical
   dx[i] -= dxi;
  }
 }
}
```

Models of parallelism