www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Tutorial OmpSs: Single node advanced examples

PUMPS 2013 tutorial
Hybrid and Heterogeneous Parallel Programming with MPI/ OmpSs for Exascale Syste

Rosa M Badia, Xavier Martorell

EXCELENCIA SEVERO OCHOA

PUMPS 2013, 12 Juliol 2013

Tutorial OmpSs

Agenda

9:00 – 10:30	Introduction to StarSs OmpSs syntax Simple examples Development methodology and infrastructure	90 min
10:30 – 11:00	Coffee break	30 min
11:10 – 12:30	Hands-on single node (I)	90 min
12:30 – 13:30	Lunch	60 min
13:30 – 15:00	Support for heterogeneous platforms Advanced examples	90 min
15:00 – 15:15	Short break	15 min
15:15 – 17:00	Hands-on (II)	105 min

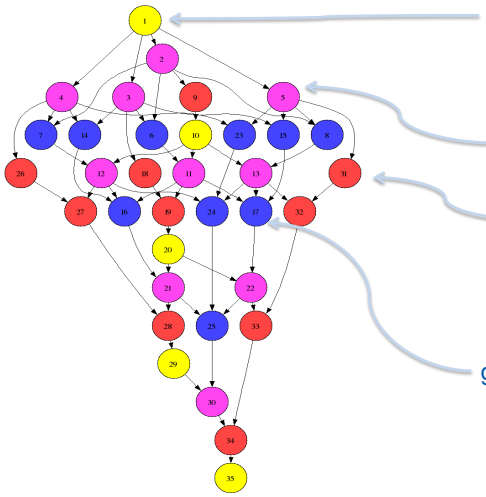
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Giving hints to the compiler: priorities

```
for (k = 0; k < nt; k++) {
        for (i = 0; i < k; i++) {
          #pragma omp task in([ts*ts]Ah[i*nt + k]) inout([ts*ts]Ah[k*nt + k]) \
          priority( (nt-i)+10 ) firstprivate (i, k, nt, ts)
            syrk_tile (Ah[i*nt + k], Ah[k*nt + k], ts, region)
        }
        // Diagonal Block factorization and panel permutations
        #pragma omp task inout([ts*ts]Ah[k*nt + k]) \
         priority( 100000 ) firstprivate (k, ts, nt)
        potr_tile(Ah[k*nt + k], ts, region)

        // update trailing matrix
        for (i = k + 1; i < nt; i++) {
            for (j = 0; j < k; j++) {
                #pragma omp task in ([ts*ts]Ah[j*nt+i], [ts*ts]Ah[j*nt+k]) \
                 inout ( [ts*ts]Ah[k*nt+i]) firstprivate (i, j, k, ts, nt)
                gemm_tile (Ah[j*nt + i], Ah[j*nt + k], Ah[k*nt + i], ts, region)
            }
            #pragma omp task in([ts*ts]Ah[k*nt + k]) inout([ts*ts]Ah[k*nt + i]) \
            priority( (nt-i)+10 ) firstprivate (i, k, ts, nt)
            trsm_tile (Ah[k*nt + k], Ah[k*nt + i], ts, region)
        }
    }
#pragma omp taskwait
```


Giving hints to the compiler: priorities



Potrf: Maximum priority

trsm: priority (nt – i) + 10

syrk: priority (nt – i) + 10

gemm: no priority

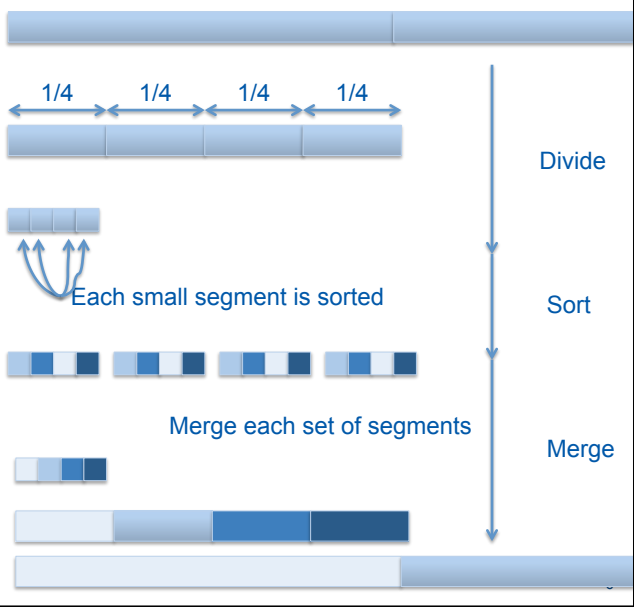
Giving hints to the compiler: priorities

- Two policies available:
  - Priority
    - Tasks are scheduled based on the assigned priority
    - The priority is a number >= 0. Given two tasks with priority A and B, where A > B, the task with priority A will be executed earlier than the one with B
    - When a task T with priority A creates a task Tc that was given priority B by the user, the priority of Tc will be added to that of its parent. Thus, the priority of Tc will be A + B
  - Smart Priority
    - Similar to the Priority, but also propagates the priority to the immediate preceding tasks
- Using the schedulers:
  - export NX_ARGS= "--schedule-priority"
  - export NX_ARGS = "--schedule-smartpriority"
- Right now only available in bf, dbf schedulers

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Accessing non-contiguous or partially overlapped regions

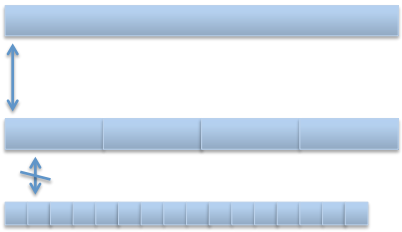
- Sorting arrays
  - Divide by ¼
  - Sort
  - Merge



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Accessing non-contiguous or partially overlapped regions

- Why is the regions-aware dependences plug-in needed?
  - Regular dependence checking uses first element as representative (size is not considered)
  - Segment starting at address A[i] with length L/4 will be considered the same as A[i] with length L
  - Dependences between A[i] with lenght L and A[i+L/4] with length L/4 will not be detected
- All these is fixed with the regions plugin
- Two different implementations:
  - **NX_DEPS=** regions
  - **NX_DEPS=** perfect-regions



Accessing non-contiguous or partially overlapped regions

```
void multisort(long n, T data[n], T tmp[n]) {
   if (n >= MIN_SORT_SIZE*4L) {
   // Recursive decomposition
   #pragma omp task inout (data[0;n/4L]) firstprivate(n)
   multisort(n/4L, &data[0], &tmp[0]);
   #pragma omp task inout(data[n/4L;n/4L]) firstprivate(n)
   multisort(n/4L, &data[n/4L], &tmp[n/4L]);
   #pragma omp task inout (data[n/2L;n/4L]) firstprivate(n)
   multisort(n/4L, &data[n/2L], &tmp[n/2L]);
   #pragma omp task inout (data[3L*n/4L; n/4L]) firstprivate(n)
   multisort(n/4L, &data[3L*n/4L], &tmp[3L*n/4L]);

   #pragma omp task in (data[0;n/4L], data[n/4L;n/4L]) out (tmp[0; n/2L])\
   firstprivate(n)
   merge_rec(n/4L, &data[0], &data[n/4L], &tmp[0], 0, n/2L);
   #pragma omp task in (data[n/2L;n/4L], data[3L*n/4L; n/4L])\
   out (tmp[n/2L; n/2L]) firstprivate (n)
   merge_rec(n/4L, &data[n/2L], &data[3L*n/4L], &tmp[n/2L], 0, n/2L);

   #pragma omp task in (tmp[0; n/2L], tmp[n/2L; n/2L]) out (data[0; n]) firstprivate (n)
   merge_rec(n/2L, &tmp[0], &tmp[n/2L], &data[0], 0, n);
   }
  else basicsort(n, data);
}
```

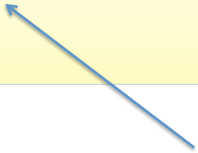
Accessing non-contiguous or partially overlapped regions

```
void merge_rec(long n, T left[n], T right[n], T result[n*2], long start, long le
ngth) {
        if (length < MIN_MERGE_SIZE*2L) {
                // Base case
                basicmerge(n, left, right, result, start, length);
        } else {
                // Recursive decomposition
                #pragma omp task in (left[0;n], right[0;n])\
                 out (result[start;length/2]) firstprivate(n, start, length)
                merge_rec(n, left, right, result, start, length/2);
                #pragma omp task in (left[0;n], right[0;n])\
                 out (result[start+length/2;length/2]) firstprivate(n, start, length)
                merge_rec(n, left, right, result, start + length/2, length/2);
        }
}
```


Accessing non-contiguous or partially overlapped regions

```
T *data = malloc(N*sizeof(T));
T *tmp = malloc(N*sizeof(T));

posix_memalign ((void**)&data, N*sizeof(T), N*sizeof(T));
posix_memalign ((void**)&tmp, N*sizeof(T), N*sizeof(T));
. . .
multisort(N, data, tmp);
#pragma omp taskwait
```



Current implementation requires alignment of data for efficient data-dependence management

Using task versions

```
#pragma omp target device (smp) copy_deps
#pragma omp task input([NB][NB]A, [NB][NB]B) inout([NB][NB]C)
void matmul(double  *A, double *B, double *C, unsigned long NB)
{
  int i, j, k, I;
  double tmp;
  for (i = 0; i < NB; i++) {
    I=i*NB;
    for (j = 0; j < NB; j++) {
      tmp=C[I+j];
      for (k = 0; k < NB; k++)
        tmp+=A[I+k]*B[k*NB+j];
      C[I+j]=tmp;
    }
  }
}
#pragma omp target device (smp) implements (matmul) copy_deps
#pragma omp task input([NB][NB]A, [NB][NB]B) inout([NB][NB]C)
void matmul_mkl(double  *A, double *B, double *C, unsigned long NB)
{
   cblas_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans, NB, NB, NB, 1.0,
   (double *)A, NB, (double *)B, NB, 1.0, (double *)C, NB);
}
```

11

Using task versions

```
void compute(struct timeval *start, struct timeval *stop, unsigned long NB, unsi
gned long DIM, double *A[DIM][DIM], double *B[DIM][DIM], double *C[DIM][DIM])
{
  unsigned i, j, k;

  gettimeofday(start,NULL);

  for (i = 0; i < DIM; i++)
    for (j = 0; j < DIM; j++)
      for (k = 0; k < DIM; k++)
        matmul ((double *)A[i][k], (double *)B[k][j], (double *)C[i][j], NB);

  #pragma omp taskwait
  gettimeofday(stop,NULL);
}
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

12

Using task versions

- Use of especific scheduling:
  - export NX_SCHEDULE=versioning
- Tries each version a given number of times and automatically will choose the best version

Using socket aware scheduling

- Assign top level tasks (depth 1) to a NUMA node set by the user before task creation
  - nested tasks will run in the same node as their parent.
- nanos_current_socket API function must be called before instantiation of tasks to set the NUMA node the task will be assigned to.
- Queues sorted by priority with as many queues as NUMA nodes specified (see num-sockets parameter).

Using socket aware scheduling

Example: stream

```
#pragma omp task in ([bs]a, [bs]b) out ([bs]c)
void add_task (double *a, double *b, double *c, int bs)
{
   int j;
   for (j=0; j < BSIZE; j++)
      c[j] = a[j]+b[j];
}

void tuned_STREAM_Add()
{
   int j;
   for (j=0; j<N; j+=BSIZE){
       nanos_current_socket( ( j/((int)BSIZE) ) % 2  );
       add_task(&a[j], &b[j], &c[j], BSIZE);
   }
}
```

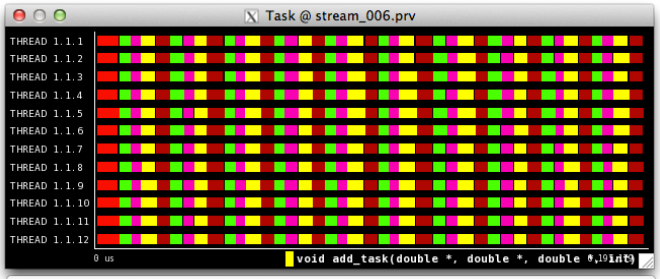

Using socket aware scheduling

- Usage:
  - export NX_SCHEDULE=socket
- If using less than 6 threads:
  - export NX_ARGS="--binding-stride 6"

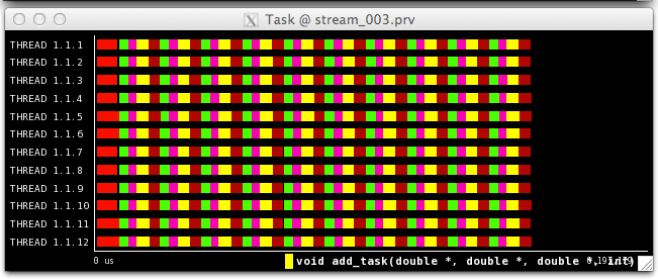
Using socket aware scheduling

Differences between the use of socket aware scheduling in the stream example:

Non Socket-aware



Socket-aware



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

17

www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Thank you!

For further information please contact
rosa.m.badia@bsc.es

18