OpenMP 3.0 Tasking Implementation in OpenUH

Cody Addison  
Texas Instruments

James (Jim) LaGrone  
University of Houston

Lei Huang  
University of Houston

Barbara Chapman  
University of Houston
Outline

• OpenUH compiler
• OpenMP and new features
• Implementing OpenMP tasks in OpenUH
  • Methodology, Frontend, RTL, and Translation
• Results
OpenUH Compiler

- OpenUH is an Open64-based open source compiler suite
- C, C++, Fortran 77/90/95
- OpenMP 2.5 compliant
Projects with OpenUH in HPCTools Group

- OpenMP 3.0 and extensions
- OpenMP Collector API
- Performance tools integration
- Parallel DFA
- Co-Array Fortran
- UPC
- Embedded OpenMP
OpenMP

- *De facto* standard for shared-memory programming
- C, C++, Fortran
- Directives, runtime library, & env. variables
- Gaining acceptance in multicore era
- Easy to incrementally add parallelism to existing code
<initialize data>

for(i = 0; i < n; ++i)
{
    <do work here>
}

OpenMP
OpenMP

#pragma omp parallel
{
  <initialize data>

#pragma omp for schedule(static,n/2)
for(i = 0; i < n; ++i)
{
  <do work here>
}
}
OpenMP 3.0

- **Tasks**
- Loop construct changes
  - allow collapsing of perfectly nested loops
  - enhanced schedules
- Improved nested parallelism support
- Improved C++ support
- Other
  - stack size control, idle thread control
OpenMP Tasks

- Simplifies parallelization of irregular algs.
  - recursion, ptr-based data structures
- OpenMP 3.0 – adds explicit tasks + sync
- A task executes at some future time
  - Execution can be tied to a thread or stolen by another
  - Execution can be suspended
- A task construct can appear anywhere
- Sync with `taskwait`
int fib(int n) {
    int x, y;
    if (n < 2)
        return n;
    else {
        #pragma omp task shared(x)
        x = fib(n - 1);
        #pragma omp task shared(y)
        y = fib(n - 2);
        #pragma omp taskwait
        return x + y;
    }
}
Fibonacci Task Graph
OpenMP in OpenUH

FRONTENDS
(C/C++, Fortran 90, OpenMP)

IPA
(Inter Procedural Analyzer)

OMP_PRELOWER
(Preprocess OpenMP)

LNO
(Loop Nest Optimizer)

LOWER_MP
(Transformation of OpenMP)

WOPT
(global scalar optimizer)

WHIRL2C & WHIRL2F
(IR-to-source for none-Itanium)

CG
(code for IA-32, IA-64, Opteron)

Source code w/ OpenMP directives

Source code with runtime library calls

A Native Compiler

Object files

Linking

Executables

A Portable OpenMP Runtime library
Frontend

- C/C++
  - Uses GCC 3.3 frontend
  - Extended to fully support OpenMP tasks
- Fortran
  - Cray fe90 frontend
  - In progress

OpenUH does not yet support GCC 4.2 F.E.
Tasking Runtime

• Task Scheduling
  • distributed, work-stealing scheduler
  • breadth-first creation, depth-first execution

• Task Synchronization
  • `taskwait` and `barrier`

• Task Switching
  • Portable Coroutines Library
    • `setjump/longjmp` or `ucontext`
Task Queues in OpenUH
Role of OpenMP Runtime

```
#pragma omp task shared(x)
  x = fib(n-1);

int fib(int n){
  ...
  __ompc_task_create( __ompc_task_1,
                      (void*)__omp_task_args_1, 1 );
  }else{
    int __omp_local_n;
    __omp_local_n = n;
    fib(__omp_local_n - 1);
  }
  ...

  return_value = pthread_create(
                         (__omp_level_1_pthread[i].uthread_id),
                         &__omp_pthread_attr,
                         (pthread_entry) __ompc_level_1_slave,
                         (void *)((unsigned long int)i));
  ...
```
Translation Methodology

- Decide on (manual) translation of construct
- Implement runtime with selected translation
  - Test & evaluate
- Modify front ends
- Implement translation
- Test, evaluate, & improve
Translation of Tasks

```c
struct omp_task_1_args_ty{
    int n;
    int *x;
};

void __ompc_task_1(void *fp){
    struct omp_task_1_args_ty *args;
    int n, *x;

    args = (struct omp_task_1_args_ty *)fp;
    n = args->n;
    x = args->x;
    *x = fib(n - 1);
    __ompc_task_exit();
}
```
Translation of Tasks

int fib(int n){
    int x, y, sum;
    if (n < 2)
        return n;

    if (__ompc_task_create_cond()){
        struct omp_task_1_args_ty *__omp_task_args_1;
        __omp_task_args_1 =
        malloc( sizeof(struct omp_task_1_args_ty));
        __omp_task_args_1 -> n = n;
        __omp_task_args_1 -> x = &x;
        __ompc_task_create(__omp_task_1,
                           (void*)__omp_task_args_1, 1 );
    }
    else{
        int __omp_local_n;
        __omp_local_n = n;
        fib(__omp_local_n - 1);
    }
    .....  
    __ompc_task_wait();
    return x + y;
}
Results

- NQueens, Multisort, SparseLU, Strassen
- Compared to
  - Cilk 5.4.6
  - Nanos 4.2 (src2src)
Thank you!

- Questions???