OpenMP
Threading on Blue Gene Q

T. Hater

JSC

February 5, 2013
Motivation

- BG/Q CPU: 16 cores with 4-way SMT: 64 threads total.
- Two instruction pipelines per core.
⇒ Need to issue two instructions per cycle and core.
Outline

1. Very fast intro to OpenMP.
2. How to use it on BG/Q.
3. Things to watch out for.
OpenMP fast forward
What is OpenMP?

- Platform independent threading standard.
- Support by most major compilers.
- Relies on programmer annotations with pragmas.
- Additional runtime functions.
Parallel regions

- Basic threading
  
  ```
  !$omp parallel
  ... executed n times ...
  !$omp end parallel
  ```

- Forks OMP_NUM_THREADS threads and joins them.

  ⇒ Implicit barrier at the end.

- OpenMP constructs appear inside parallel.
Simple loops

- Parallelize a counting loop

  ```
  !$omp parallel do
  do i = 1,N
  ... 
  end do
  !$omp end parallel do
  ```

- Every thread is assigned a chunk of the iterations.

- May specify scheduling
  ```
  schedule({static|dynamic|guided}[,, chunk_size]).
  ```
  ⇒ runtime leaves it to OpenMP.

- collapse merges nested loops.
parallel construct may include access attributes for variables

\[
!\text{omp parallel clause}(\text{var}, \ldots) \\
\ldots \\
!\text{omp end parallel}
\]

Where clause is one of \{\text{shared}|\text{private}|\text{firstprivate}\}.
Special: reduction(\text{var:op}, \ldots).
Set the default: default(\{\text{shared}|\text{none}\})
Synchronization

- Only a single thread at a time.
  \[
  \texttt{!omp critical} \quad \ldots \text{one at a time} \ldots \quad \texttt{!omp end critical}
  \]

- Only a single (the master) thread.
  \[
  \texttt{!omp \{master|single\}} \quad \ldots \text{once} \ldots \quad \texttt{!omp end \{master|single\}}
  \]

- Atomic operations.
  \[
  \texttt{!omp atomic}
  \]

- Wait for all threads.
  \[
  \texttt{!omp barrier}
  \]
Sections

```c
!$omp sections
!$omp section
    ... one thread ...
!$omp end section
...
!$omp section
    ... another thread ...
!$omp end section
!$omp end sections
```

- Does not scale!
Tasks

- Create a task
  \[ !\text{omp task} \quad \text{... concurrently ...} \quad !\text{omp end task} \]

- One task per encountering thread

- Synchronizing tasks
  \[ !\text{omp taskwait} \]

⇒ Only for \textbf{direct} descendants.
OpenMP on Blue Gene Q
Compiling for OpenMP

- Using the XL compilers.
- Add `-qsmp=omp` to compiler and linker flags.
- Automatically enables `-O2 -qhot`.
  ⇒ Suppress with `-qsmp=omp:noopt`.
- XL can try to automatically parallelize loops.
  ⇒ Enable on top of OpenMP with `-qsmp`.
XL OpenMP

- OMP_NUM_THREADS = OMP_THREAD_LIMIT = \( \frac{64}{\text{RanksPerNode}} \)
- May oversubscribe, but be careful.
- OMP_PROC_BIND = True (fixed) due to CNK limitation.
- No conforming nested OpenMP.
- Utilize thread local storage with !IBM* THREADLOCAL.
Exploiting BG/Q features

- XL OpenMP runtime already uses some BG/Q features.
- `omp barrier` and `lock` use L2 atomic hardware support.
- `omp atomic` exploits hardware atomic support.
- Waiting threads go to sleep.
Tuning for performance
General ideas

- Try threaded libraries.
- Combine multiple constructs into one region.
- Cut down on synchronization.
- Consider `nowait`.
- Avoid `flush`. 
Overhead

- Rule of thumb: 100µs for tasks, 10µs for loops and 1µs else.
  - \{lock|barrier|critical\} \leq 1µs
  - parallel 1µs (1 thread) − 50µs (64 threads)
  - do loops 1µs (1 thread) − 50µs (64 threads)
  - Task create/wait 2µs (1 thread) − 50µs (16 threads)
- Scheduling: prefer runtime and static.
Mixing OpenMP and MPI

- Good starting point: 16 MPI processes + 4 OpenMP threads
- Issues
  - Global parallelization vs hotspots
  - Explicit communication vs shared memory issues
  - Memory usage
Resources

- XL compiler manuals
  http://pic.dhe.ibm.com/infocenter/compbg/v121v141/

- OpenMP standard

- OpenMP overview card
Questions?
MPI Comm threads

- If unused threads are available, BG/Q can use them to asynchronously make progress on MPI requests.
- To use, initialize MPI with `MPI_Init_thread` and `MPI_THREAD_MULTIPLE`.
- The rest should happen automagically.

⇒ Not tested.
Tuning worksharing

- Control the runtime via environment variable
  \texttt{XLSMPOPTS="key=val : ..."}

- Basic: specify defaults for OpenMP attributes

- Advanced: Tune work sharing
  - \texttt{yields[num], spins[num] and delays[num].}
  - Algorithm for dynamic worksharing.
    1. Scan for work spins times, if nothing, idle for delays.
    2. Scan again, if still nothing yield to another thread.
    3. Repeat yields times, then go to sleep.
  - Set \texttt{spins=yields=0} to force pure busy waiting.

- Set \texttt{BG\_SMP\_FAST\_WAKEUP=YES}.

⇒ Might be useful, but I found no gain.

⇒ Be wary of deadlocks if oversubscribing.