Performance Analysis of and Tool Support for Transactional Memory on BG/Q

Workshop on Transactional Memory
April 10, 2012

Martin Schindewolf\textsuperscript{12}, Martin Schulz\textsuperscript{1}, Barna Bihari\textsuperscript{1}, John Gyllenhaal\textsuperscript{1}, Amy Wang\textsuperscript{3}, Wolfgang Karl\textsuperscript{2}
Motivation

- BG/Q is installed as Sequoia at LLNL
  - 20 PFLOP/s
  - 1.6 million cores
  - Each with 4-way hyper threading
  - >6 Million threads may run in parallel

- Hybrid parallelization will be the key to performance
  - Need to use MPI between nodes
  - Combine with OpenMP/Pthreads on node level
  - Efficient synchronization mandatory for performance
Motivation

- BG/Q has support for
  - Transactional Memory
  - Speculative Execution
- (How) does TM work?
- Study performance of TM
  - How/when to use it?
    - Compare with other synchronization mechanisms
    - Influence of environment variables
  - Combine TM with MPI?
- How can tools for TM look like?
Outline

- Transactional Memory
  - Concept
  - Support on BG/Q
- Correctness testing with tm rand
- Performance and overhead analysis
  - CLOMP-TM
  - CLOMP-TM with MPI
- Tool design and implementation
- Conclusion
TM Concept

- Transactions guarantee
  - Atomicity
  - Consistency
  - Isolation

- Optimistic concurrency
  - Detect and resolve conflicting accesses
  - One transaction needs to roll back its changes
Support for TM on BG/Q

- HW support for conflict detection
- If multiple Txns conflict, all are aborted
- Register save and restore is done in SW
- No isolation in irrevocable mode
- Prototype of IBM XL C/C++ and Fortran compilers provide the compiler and runtime support needed to use TM and SE
TM Programming Model

- Programmer defines transaction in the code

```c
#pragma tm_atomic {
    account1 -= amount;
    account2 += amount;
}
```

- Transaction are single-entry and single-exit code blocks
  - No ISA limitation
  - Txn boundaries must be determinable statically by compiler
  - Control flow that exits Txn may result in compile- or runtime error
  - Jail mode protects the system from potential ‘speculative’ damage (e.g., I/O)
Correctness Testing

- Test case tm_rand:
  - Array of counters
  - Transactional update of "own" or "foreign" counter
  - Conflict probability decides which counter to update
  - Results show:
    - TM behavior matches expectations in most cases
    - TM detected and resolved all conflicts correctly for CLOMP-TM and tm_rand

- Mapping counter array to the cache lines
  - Multiple counters in one line
  - Classify accesses as conflicts -> False Positives
  - Avoid through padding of the counter array
Correctness Testing cont’d (2)

<table>
<thead>
<tr>
<th>Intended conflicts</th>
<th>Serialized, pad 64</th>
<th>Retries, pad 128</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retries, no pad</td>
<td>Retries, pad 128</td>
<td></td>
</tr>
<tr>
<td>Serialized, no pad</td>
<td>Serialized, pad 128</td>
<td></td>
</tr>
<tr>
<td>Retries, pad 64</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

tm_rand with 4 threads, RBM 1 on BG/Q
Correctness Testing cont’d (3)

Intended conflicts
Retries, no pad
Serialized, no pad
Retries, pad 64
Retries, pad 128
Serialized, pad 64
Serialized, pad 128

*tm_rand* with 16 threads, *RBM 1* on *BG/Q*
Performance Analysis – CLOMP-TM

- Parameter set defines:
  - Parts, Zones, scatterZones,
  - ScatterMode defines contention
    - None, Adjacent, Random, firstParts, and mixed modes
  - Configurable amount of computation per zone update
    - none, divide, complex
    - Scaled by a factor

- Compare different synchronization mechanisms
  - Quantify overheads
  - Determine best length
Performance Analysis – CLOMP-TM
Performance Analysis – CLOMP-TM
CLOMP-TM with MPI Experiments

- Add MPI_Barrier construct
  - All MPI tasks execute one thread synchronization mechanisms at the same time
  - Increases pressure on architecture
- Determine best task-to-thread ratio

```c
MPI_Barrier(MPI_COMM_WORLD);
get_timestamp (&bestcase_start_ts);
do_bestcase_version();
get_timestamp (&bestcase_end_ts);
MPI_Barrier(MPI_COMM_WORLD);
```
CLOMP-TM with MPI (no cont.)
CLOMP-TM with MPI (high cont.)

![Graph showing speedup over serial against #MPI tasks for different scenarios: Bestcase, Large TM, Small Critical, Small TM, Small Atomic, and Large Critical.](image)
Tool Support for TM

- Library implementation
- Idea: correlate TM stats and BGPM performance counter values
- Design goals
  - Simple API
  - Transparent for User
  - Hide interface of TM run time and BGPM
Conclusion

- No errors found in HW
- TM should be used when
  - Contention is low (few rollbacks)
  - Coarse grain Txns are possible
  - omp atomic is not applicable
  - L2 is not yet exhausted
- Tool support for TM is ongoing work
  - Prototype is implemented and tested