COMIC: A Coherent Shared Memory Interface for Cell BE

Jaejin Lee
Advanced Compiler Research Laboratory
School of Computer Science and Engineering
Seoul National University
http://aces.snu.ac.kr

June 4, 2008
Cell BE Architecture

- **PPE**
  - A PowerPC core and for general purpose processing
- **8 SPEs**
  - To accelerate media and streaming workloads
  - Access the main memory through DMA transfers to the local store
- Communication between the PPE and SPEs
  - Mailboxes and signals
- No coherence between the local stores or between the local stores and the external memory
  - Hard to program
Coherence

- SPE0 reads \( x \) in the main memory
- SPE1 reads \( x \) in the main memory
- SPE0 writes to the local copy of \( x \)
- Stale copies of \( x \) in SPE1 and the main memory
COMIC

- A runtime system
  - API functions
- A globally shared address space between PPE and SPEs
  - An SPMD-style parallel programming model
- An efficient software cache in the local store for page level caching
  - Guarantees coherence at the page level
  - Centralized lazy release consistency
Thread Model

PPE thread

SPE threads

- servicing SPE thread
- performing a task
- waiting for a task
- task assignment
Cell BE Code Generation

- app.c
  - programmer or compiler
  - app_c
    - SPE compiler
    - app_spec
    - SPE linker
    - SPE executable
    - SPE embedder
    - SPE object
    - PPE object
    - PPE linker
    - Cell BE executable
  - app_ppe_c
    - PPE compiler
    - app_ppe_o
    - PPE linker
    - Cell BE executable

- COMIC_lib_SPE.so
- COMIC_lib_PPE.so
COMIC Software Cache

- In the local store of each SPE
- Each read or write in the SPE code is replaced by a COMIC software cache read or write function
- For page-level caching
COMIC Program

(a) OpenMP

(b) PPE code

(c) SPE code

#include <stdio.h>
#include <omp.h>
#define SIZE 24
int a[ SIZE ];
int main() {
    int i;
    #pragma omp parallel for
    for( i=0; i < SIZE; i++ )
        a[i] = i;
    for( i=0; i < SIZE; i++ )
        printf( "\%d\n", a[ i ] );
    return 0;
}

#include <stdio.h>
#include <omp.h>
#define SIZE 24
int a[ SIZE ];
int main() {
    int i;
    COMIC_ppe_init();
    COMIC_ppe_run( 0 );
    for( i=0; i < SIZE; i++ )
        printf( "\%d\n", COMIC_access( a[ i ] ) );
    COMIC_ppe_exit();
    return 0;
}

#include <stdio.h>
#define SIZE 24
COMIC_shared ( int a[ SIZE ]; )
COMIC_ppe_main ( case 0: {
    int i, tid, chunk, low, high;
    tid = COMIC_get_spe_thread_id();
    chunk = SIZE / COMIC_MAX_NUM_SPE_THREADS;
    low = tid * chunk;
    high = (tid == (COMIC_MAX_NUM_SPE_THREADS-1)) ? SIZE : (low + chunk);
    for( i = low; i < high; i++ )
        COMIC_access( a[ i ] )
            COMIC_ppe_write_int( &a[i], i );
    break;
} )
Evaluation

- Compared to XL C/C++ Alpha edition single-source OpenMP compiler from IBM alphaWorks
- 12 OpenMP applications from NAS, SPEC OMP, HPC Challenge, and www.OpenMP.org
Cost/Performance Trade-Off

- Compared to an Intel Xeon quadcore two-socket server (total 8 cores)
  - The server is 1.52 times faster than COMIC with 1166M transistors in the two Xeon quadcore chips
  - A single Cell chip contains 234M transistors, i.e., 4.98 times cheaper than the two Xeon chips
- COMIC achieves 3.23 times better performance than the Xeon server with the same transistor cost
Conclusion

- Programming COMIC is easier than programming the raw Cell BE hardware
- COMIC achieves high performance
- The transistor-cost/performance trade-off of COMIC system is better than SMP-like homogeneous multicores
- COMIC is a good target of compilers for higher-level programming models (UPC, OpenMP, and the like)