New Approaches for Scalable Software on CMPs

CASCON 2006 Workshop
Challenges for Parallel Computing

Xavier Martorell

Computer Architecture Dept. (AC)
Technical University of Catalunya (UPC)
Barcelona Supercomputing Center (BSC)
Outline

- ScalAble ComputeR ArchiteCture (SARC)
- Programming model approaches
- Runtime system support
- Experiments
- Conclusions
- Future work
SARC

• Funded by the Information Society Technologies (IST) of the EU
  – 4 years project, started Jan 2006

• Partners
  – Delft University of Technology (coord)
  – Chalmers University
  – Edinburgh University
  – FORTH
  – Ghent University
  – INRIA
  – ISI
  – University of Pisa
  – Barcelona Supercomputing Center
  – Universidad Politécnica de Valencia
  – ARM UK
  – IBM Israel
  – Philips
  – ST Microelectronics
  – Thales
  – Xyratex
SARC

• Workpackages
  – WP1, Scalable multi-core processors
  – WP2, Scalable memory systems
  – WP3, Adaptive and global optimization
  – WP4, Scalable switches and networks
  – WP5, Processor - network interface
  – WP6, Programming model and run-time sys...
  – WP7, Design space exploration
SARC Architecture

● Communication centric
  - Reconfigurable to application needs

● Domain centric
  - Targeting specific applications

● New memory organization
  - Reduction of traffic
  - Reconfigurable
Target applications

• Supercomputing
  – CPMD
  – NAMD
  – LISO

• Bio-informatics
  – Amber
  – BLAST
  – HMMER

• Multimedia
  – H.264/AVC
  – MPEG4
SARC Architecture

- Prototype CMP
  - Accelerators
  - Local caches
  - Local memories
  - DMA engines

- Research
  - Design of units
  - Evaluate different ways of connecting the elements

[Diagram of SARC Architecture showing CPU, Cache, Accelerators, Local Store, DMA, and Memory with inter-block and intra-block networks]
Programming model

• Mainly based on OpenMP
  – OpenMP provides the tools
  – It can be extended further to CMPs
  – Because it is where we have experience

• Exploit component-based programming
  – Generate parallelism also by splitting objects in OO environments (C++)

• Model accelerators like spawning code through remote procedure calls
Runtime system

• Build an actual prototype (we always do)
  – Search for the appropriate platforms
    • Multicore Pentium/Power/Itanium
    • Cell BE

• Build on our existing runtime system
  – Nanos, from past european projects
    • Nanos
    • Intone
    • POP
OpenMP for Cell?

• Right, OpenMP is not for distributed memory but it allows to easy express the parallelism

• Work to do
  – Outline SPUs code
  – Find input and output parameters needed for the functions outlined to the SPUs, find other functions called
  – Have an “idle” code running at the SPUs, searching for work
OpenMP for Cell?

- Approach
  - Starting from our current OpenMP
  - Idea: Let the PUs define the work and the SPU{s} execute it
    - No decisions taken at the SPU{s}
OpenMP for Cell

• Example (from NAS MG)
  - Parallel
  - Input
  - Output
  - Input by value

```plaintext
do j3=2,m3j-1
  i3 = 2*j3-d3
  do j2=2,m2j-1
    i2 = 2*j2-d2
    x1(i1-1) = r(i1-1,i2-1,i3  ) + r(i1-1,i2+1,i3  )
    y1(i1-1) = r(i1-1,i2-1,i3-1) + r(i1-1,i2-1,i3+1)
  enddo
  do j1=2,m1j-1
    i1 = 2*j1-d1
    s(j1,j2,j3) = ...
  enddo
enddo

!$cel parallel do private (i1, j1, ...)
!$cel& input (r) from ( 1:+1, -1:+1, ...)
!$cel& output (s)
!$cel& copyin (m1j, m2j, m3j...)
!$cel& ...
```

Not for beginners
OpenMP for Cell

• ... but it helps developers (us) to understand what's going on

• Initial experiments
  – Matrix addition
  – Matrix product
Experiments

• Matrix addition example

```c
!$omp parallel
!$cel do schedule (static, 2)
!$cel& private (i)
!$cel& allocate (A) input (A) from (LOOP_START : LOOP_END)
!$cel& allocate (B) input (B) from (LOOP_START : LOOP_END)
!$cel& allocate (C) output (C) from (LOOP_START : LOOP_END)
    do j = 1, N
        do i = 1, N
            C(i,j) = A(i,j) + B(i,j)
        enddo
    enddo
!$omp end parallel
```

– Schedule clause helps with the amount of data brought in/out !!!

• Scheduling decisions left for PU!
• But... overhead?
Experiments

- Matrix addition, single precision, no vectorization

Cell performance
matrix addition (2048x2048), single precision

Execution time (microsecs.)

0 25000 50000 75000 100000 125000 150000 175000 200000 225000

Serial PU Serial SPU Parallel PU Parallel SPU

1 thrd 2 thrs 3 thrds 4 threads

Barcelona Supercomputing Center
Centro Nacional de Supercomputación
Experiments

- Matrix addition, double precision
  no vectorization

Cell performance
matrix addition (1024x1024), double precision

PU is not good for work,
it is good to distribute work!!
Experiments

• Open questions
  – Vectorization
  – Computation/communication overlap?
    • Double buffering
  – Automatic code partitioning & overlays?
  – Getting more information on what's going on?
    • On which SPU's am I executing?
    • Are they on the same Cell chip? Across chips?
Experiments

• Matrix multiply:
  
  - New clause: multioutput or scattering data

```c
!$omp parallel
!$cel do private(i,k,tmp)
!$cel& allocate (A) input (A) ! all of A is copied
!$cel& allocate (B) input (B) from (0:0, LOOP_START : LOOP_END)
!$cel& allocate (C) multioutput (C)
!$cel& from (LOOP_START : LOOP_END, 0:0)
!$cel& schedule(static,4)
    do j = 1, N
      do i = 1, N
        tmp = 0.0d0
        do k = 1, N
          tmp = tmp + A(i,k)*B(k,j)
        enddo
        C(j,i) = tmp
      enddo
    enddo
!$omp end parallel
```
Experiments

- Scattering data

  - Consecutive memory inside local store
  - Separate columns at main memory
Experiments

• Code generated for the PU

.....
whoami = nthf_whoami()
low = 1
upper = N
step = 1
schedule = 0
chunk = 8
CALL in__tone_begin_for(low, upper, step, chunk, schedule)
DO WHILE (in__tone_next_iter(intone_start, intone_end, intone_last) /= 0)
  v_nargs = 5 + 0
  nargs = 3
  CALL spawn_to_spu(v_nargs, nargs, n_outlined_nth_matmt_cell1, whoami,
                   intone_start, intone_end, step, A, B, C)
END DO
barrier = 1
CALL in__tone_end_for(barrier)
.....
Experiments

• Code generated for the SPU

```fortran
SUBROUTINE outlined_nth__matmt_cell1
  (ol__whoami, ol__start, ol__end, ol__step, A, B, C)
...
REAL , ALLOCATABLE , DIMENSION (:, :)::fp_A, fp_B, fp_C
CALL cellf_set_alignment(A)
ALLOCATE (fp_A(LBOUND(A, DIM= 1):UBOUND(A, DIM= 1),
    LBOUND(A, DIM= 2):UBOUND(A, DIM= 2)))
CALL cellf_asynch_get(fp_A, A ( LBOUND(A, DIM= 1), LBOUND(A,DIM= 2),
    kind ( A ) * SIZE ( fp_A), 0)
CALL cellf_wait(0)
...
DO p_j = ol__start, ol__end, ol__step
  DO p_i = 1, N
    p_tmp = 0.0d0
    DO p_k = 1, N
      p_tmp = p_tmp + fp_A ( p_k, p_i ) * fp_B ( p_k, p_j )
    END DO
    fp_C ( p_i, p_j ) = p_tmp
  END DO
END DO
...
```
Experiments

• Single precision, matrix size 64x64, no vectorization
  - Good news: SPUs scale
  • Even with no computation/communication overlap
Experiments

- Double precision, matrix size 128x128
- No vectorization
  - The bigger we can do because of matrix A
  - PU better than SPU
Experiments

- Single precision 192x192 matrices, no vectorization
  - Good news, better scalability using bigger matrices
  - Even without transposition

![Algorithm Execution time (microsecs.)]
Experiments

- Single precision, 192x192, no vectorization
- Comparison against ppc970
  - 1-4 SPUs equivalent to a ppc970
Outline

• ScalAble ComputeR ArchiteCture (SARC)
• Programming model approaches
• Runtime system support
• Experiments
• Conclusions
• Future work
Conclusions

- SARC is a European project working on CMPs
  - Architecture
  - Memory hierarchy
  - Communications
  - OpenMP-like parallelization
  - Infrastructure to support easy-of-use parallelism
  - Optimizations
Conclusions

- Developing a programming model to help in the parallelization
- Current prototype of run-time system on Cell
- Experiments show it works... but much more work is needed
Future work

• Goals
  – improve the run-time infrastructure
  – research on ways of exploiting parallelism
    • Provide feedback to the architecture design
  – simplify programming / increase productivity

• OpenMP and streaming?
Acknowledgments

• Thanks to the SARC partners, specially those involved in the definition of the architecture and programming model
  – TU Delft
  – INRIA
  – FORTH
  – THALES

• This work has been supported by the European Commission in the context of the SARC integrated project #27648 (FP6)