



BlockLib: A Skeleton Library for Cell Broadband Engine™

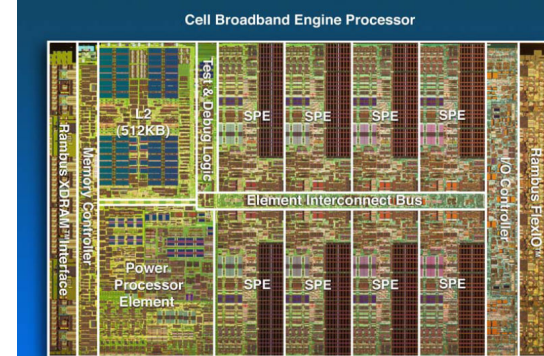
Markus Ålind, Mattias V. Eriksson, Christoph Kessler

PELAB
Linköping university
Sweden

IWMSE-2008, Leipzig, 11/5/2008



CELL BE Processor (IBM, Sony, Toshiba)

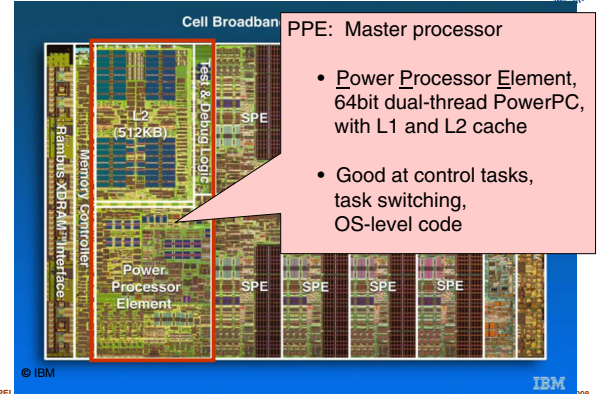


© IBM

IBM 208



CELL BE Processor (IBM, Sony, Toshiba)

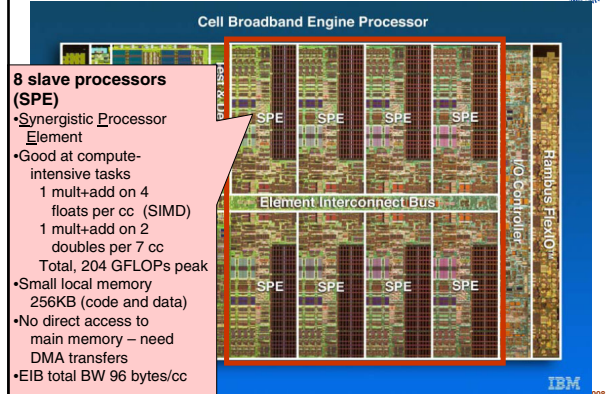


© IBM

IBM 208



CELL BE Processor (IBM, Sony, Toshiba)

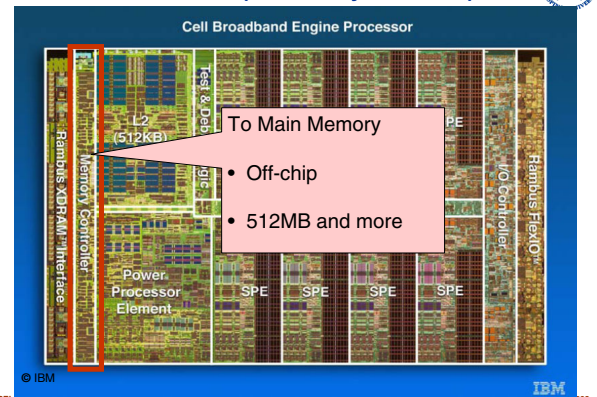


© IBM

IBM 208



CELL BE Processor (IBM, Sony, Toshiba)



© IBM

IBM 208



Issues in writing efficient SPE code

- C, IBM Cell SDK
- Local store
 - 256 kiB
- Explicit DMA transfers
 - Between main memory and local store
 - Between SPEs
 - Asynchronous
 - 16 byte aligned to work at all, 128 byte aligned to be fast
 - Multi buffering
- SPE Calculations
 - 16 byte internally
 - Loads and stores are always 16 byte
 - Vector of 4 floats faster than single scalar

PELAB, IDA, Linköping university,

6

2008

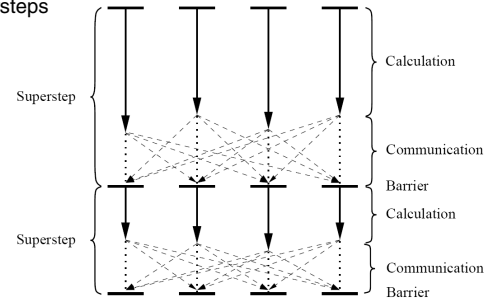
Skeleton programming

- **Skeletons**
 - For common computation patterns
 - for which efficient parallel implementations may exist
 - ▶ Data-parallel skeletons: e.g. map, reduce, scan, map-with-overlap
 - ▶ Task-parallel skeletons: e.g. farm, pipe, while
 - Generic
 - Compositional
- **Goals:**
 - Abstraction
 - ▶ Hide implementation
 - ▶ Hide parallelization
 - "Parallel programming as easy as sequential programming"
 - Code reuse

PELAB, IDA, Linköping university. 7 2008

BSP model

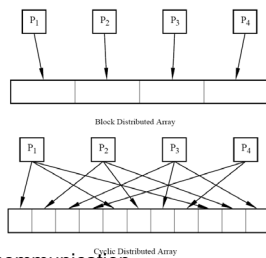
- Bulk-synchronous parallelism
- Supersteps



PELAB, IDA, Linköping university. 8 2008

NestStep

- Global address space language for the CRCW BSP model
- Locality-aware shared memory abstraction for distributed memory systems
 - Shared variables and arrays
 - Distributed shared arrays (a la HPF, UPC, ...)
 - Private variables and arrays
- Programmable combining at the end of each superstep
- Runtime system
 - manages threads, memory, communication
 - Implemented for MPI clusters and CELL BE



PELAB, IDA, Linköping university. 9 2008

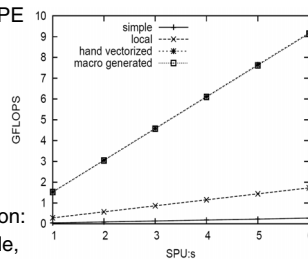
BlockLib

- Set of data-parallel skeletons
 - Map
 - ▶ $\text{map op}(x_1, \dots, x_n) := (\text{op}(x_1), \dots, \text{op}(x_n))$
 - ▶ Also with multiple argument vectors
 - Reduce
 - ▶ $\text{reduce op}(x_1, \dots, x_n) := x_1 \text{ op } x_2 \text{ op } \dots \text{ op } x_n$
 - Map-Reduce
 - Map-with-overlap
 - ▶ $\dots = (y_1, \dots, y_n)$ where $y_i := \text{op}(x_{i,k}, \dots, x_{i+k})$ for all $i = 1, \dots, n$
- Each skeleton instance (call) can be a stand-alone superstep
- or called within a superstep by a SPE thread group

PELAB, IDA, Linköping university. 10 2008

Realization 1: Generic functions

- Code parameter `op` passed via function pointer
 - Very high overhead on SPE
- Improvement 1a: User provided inner loop
- Improvement 1b: User hand SIMD optimizing
- Minor code bloat
 - Array Elements Summation: 4 lines in normal C code, 21 lines SIMD-optimized,
- Function definition - automatic optimization



PELAB, IDA, Linköping university. 11 2008

Realization 2: Generative approach

- Define code operand `op` as sequence of predefined macros
 - one per elementary arithmetic operator in op and base type (float, double)
 - easily extensible
 - expand to SPE special instructions where possible
- Code generation macros for integrated skeleton function generation
- Macro expansion by C preprocessor

```
// standard C
int i;
float sum = 0;
for (i=0; i<N; i++)
    sum += x[i]

// Definition
DEF.REDUCE_FUNC.S(my_sum, t1, BL.NONE,
    BL.SADD(t1, op1, op2))
// Usage
res = my_sum(x, N);
```

PELAB, IDA, Linköping university. 2008

BlockLib Implementation Details



- Inter-SPE communication with DMA
 - Special signal register
 - Message passing
- Synchronisation
 - Group synchronisation
 - Data combine

Evaluation



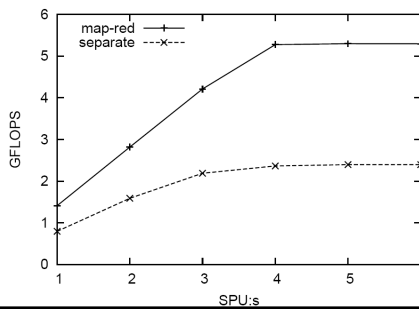
- Synthetic benchmarks
- Real application: Parallel ODE solver
 - Usability
 - Absolute performance
- Playstation 3
 - 6 SPEs
 - 210 MiB RAM
 - Vector size 5M elements each



Micro-Benchmark



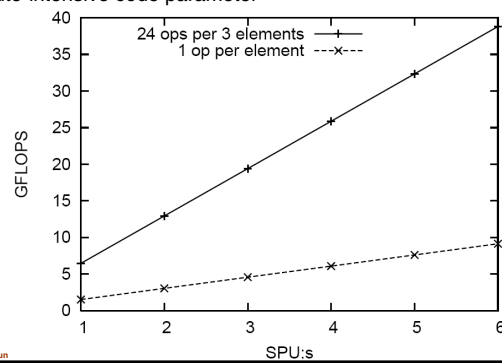
- Dot product (float)
 - 5.3 GFLOPS = 21.2 GB/s



Micro-Benchmark



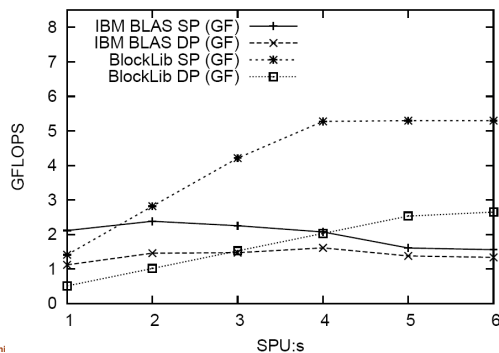
- Compute-intensive code parameter



Comparison to IBM Cell BLAS-1 Library



- Dot product



ODE Solver application



- Vector based scientific program
- Based on LibSolve by Matthias Korch and Thomas Rauber (Univ. Bayreuth)
- Brusselator equations
 - Spatial discretization of partial differential equations by the method of lines
 - Limited access distance - fits **map-with-overlap**
 - The rest is simple loops and reductions - fits **map**, **reduce** or **map-reduce**
- Double precision

ODE code excerpt, using BlockLib

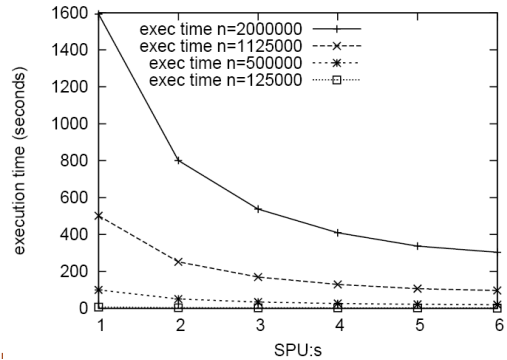
```
// original code
error_max = 0.0;
for (i = 0; i < ode_size; ++i)
{
    temp = fabs(err_vector[i] / yscal[i]);
    if (temp > error_max)
        error_max = temp;
}
// BlockLib definition
DEF_MAP_TWO_AND_REDUCE_FUNC.D(dDistMaxAbsQuot,
    abs,
    max,
    BL_NONE,
    BL_DDIV(div, m_op1, m_op2)
    BL_DABS(abs, div),
    BL_DMAX(max, r_op1, r_op2));
// usage
error_max = dDistMaxAbsQuot(err_vector, yscal, ode_size);
```

PELAB, IDA, Linköping university.

19

2008

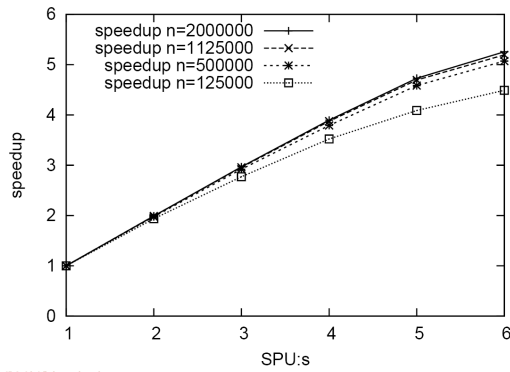
ODE Solver Execution Times



PELAB, I

2008

ODE Solver Speedup



PELAB, IDA, Linköping university.

21

2008

ODE Solver, Sources of Overhead

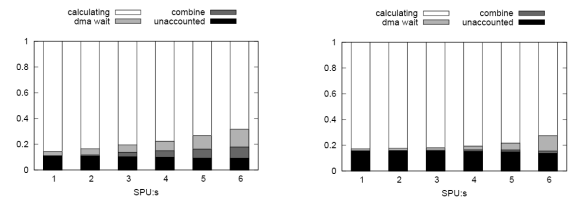


Figure 11: Time distribution of ODE solver with ode size 125000.

Figure 10: Time distribution of ODE solver with ode size 2000000.

PELAB, IDA, Linköping university.

22

2008

ODE Solver, Comparison

- Execution times for the minimal sequential solver on x86 (optimized) and the Cell port (using six SPEs)

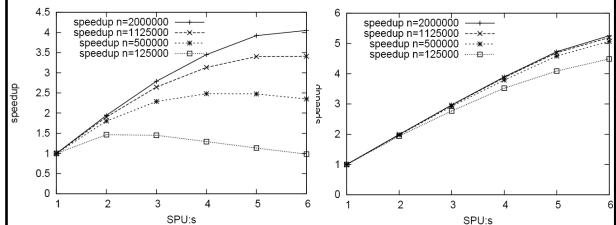
| machine | execution time |
|--------------------|----------------|
| P4 (K,R) | 2198 |
| Opteron (K,R) | 1979 |
| P4 Xeon, GCC | 1626.17 |
| P4 Xeon, ICC | 1520.23 |
| Core 2, GCC | 1386.56 |
| Cell, 6 SPE:s, GCC | 303.14 |

PELAB, IDA, Linköping university.

23

2008

ODE Solver, Synchronization Cost



Speedup using NestSteps native superstep synchronization

Speedup using BlockLib synchronization

PELAB, IDA, Linköping university.

24

2008

BlockLib Summary



- Skeleton programming library for Cell
 - Abstraction from Cell SPE programming complexity
- Called from NestStep programs or used stand-alone
- Data-parallel skeletons implemented:
 - map, reduce, map+reduce, map-with-overlap
- Variants
 - Generic functions
 - optionally with user-supplied optimized kernels
 - Generative approach
- Outperforms IBM Cell SDK 3.0 BLAS-1 dot product for $p > 2$
- Good speedup on ODE solver application

PELAB, IDA, Linköping university.

25

2008

References



- Markus Ålind, Mattias V. Eriksson, Christoph W. Kessler: BlockLib: A Skeleton Library for Cell Broadband Engine. Proc. 1st ACM SIGSOFT / IEEE Int. Workshop on Multicore Software Engineering (IWMSE-2008), Leipzig, Germany, May 2008.
- Markus Ålind: Master thesis, IDA, Linköpings universitet. Finished 2008, to appear.

PELAB, IDA, Linköping university.

26

2008