High Performance Combinatorial Algorithm Design on the Cell/B.E.

David A. Bader, Virat Agarwal
Kamesh Madduri, Seunghwa Kang, Sulabh Patel
Cell System Features

- **Heterogeneous multi-core system architecture**
  - Power Processor Element for control tasks
  - Synergistic Processor Elements for data-intensive processing

- **Synergistic Processor Element (SPE)** consists of
  - Synergistic Processor Unit (SPU)
  - Synergistic Memory Flow Control (MFC)
  - Data movement & synchronization
  - Interface to high-performance Element Interconnect Bus
Cellbuzz @ Georgia Tech.

- List ranking
- Fast Fourier Transform
- Zlib Compression/Decompression
- RC5 Encryption/Decryption
- MPEG2

Open-source, can be obtained from:
http://sourceforge.net/projects/cellbuzz
List Ranking on Cell

• Cell performs well for applications with predictable memory access patterns [Williams et. al. 2006]

• Conjecture: Can Cell architecture also perform well for applications that exhibit irregular memory access patterns?
  – Non-contiguous accesses to global data structures with low degrees of locality

List ranking is a special case of Parallel Prefix where the values are initially set to 1 (except for the head) and addition is used as the operator.
A parallel algorithm for List Ranking

- SMP algorithm [Helman & JaJa, 1999]

1. Partition the input list into $s$ sublists, by randomly choosing $s$ sublist head nodes, one from each memory block of $n/(s - 1)$ nodes.
2. Traverse each sublist computing the prefix sum of each node within the sublists.
3. Calculate prefix sums of the sublist head nodes.
4. Traverse the sublists again, summing the prefix sum values of each node with the value of its sublist head node.

- Design Issues
  - Frequent DMA transfers required to fetch successor elements.
  - No significant computation in the algorithm, thus communication creates a bottleneck.
  - Need to hide DMA latency by overlapping computation with communication.
A Generic Latency-hiding technique

- Cell supports non-blocking memory transfers

- Requires identification of another level of parallelism within each SPE.

- Concept of **software-managed-threads (SM-Threads)**
  - SPE computation is distributed among these threads
  - SM-Threads are scheduled according to Round Robin policy.

- Instruction level profiling to determine the minimum number of SM-Threads needed to hide latency.
  - Tradeoff between latency and number of SM-threads.
List Ranking: Performance Analysis

**Tuning the DMA parameter**

**Running Time (PPE-only vs PPE+SPE’s)**

![Graph showing running time and improvement factor for ordered lists.]

- **Ordered List**
  - Improvement factor: 3
  - Number of DMA buffers: 1, 2, 4, 8

![Graph showing running time and improvement factor for random lists.]

- **Random List**
  - Improvement factor: 3
  - Number of DMA buffers: 1, 2, 4, 8

**Listranking on Cell - Ordered Lists**
- PPE-only vs Cell Optimized

**Listranking on Cell - Random Lists**
- PPE-only vs Cell Optimized

Virat Agarwal, STI Cell workshop
List Ranking: Performance Analysis

Comparison with other architectures

2.5 times faster than an optimized parallel implementation on a dual core Woodcrest (Intel Xeon 5150)  
- 4.6 times faster than single core.
ZLIB: Data compression/decompression

- LZ77 algorithm [J. Ziv & A. Lempel, 1977]
  - Identify the longest repeating string from the previous data.
  - Replace the duplicated string with the reference to its previous occurrence.
  - A reference is represented by a length-distance pair.
  - Length-distance pairs and literals produced by LZ77 algorithm are Huffman coded to enhance the compression ratio.
ZLIB: Optimization for the Cell

• Optimizing on the SPE (most compute intensive parts)
  – Compression
    • Finding longest matches in LZ77 algorithm
  – Decompression
    • Converting Huffman coded data to length-distance pairs
  – Reduce memory requirement

• Parallelizing for the SPEs
  – Full flushing to break data dependency
  – Work queue to achieve load balancing
  – Extending Gzip header format to enable faster decompression
    • Include information on flush points
    • Keep it compatible with legacy gzip decompressor
GZIP Performance results

Speedup of gzip compression
Cell optimized vs sequential gzip
for a single SPE

Speedup of gzip decompression
Cell optimized vs sequential gzip
for a single SPE

Virat Agarwal, STI Cell workshop
GZIP Performance results

Performance Comparison of Cell optimized gzip with other Architectures

- Cell Optimized
- Intel 3.2 Ghz

Speedup of Cell optimized gzip compression with varying number of SPEs

- Obtained Speedup

Running Time (sec)
- 0
- 100
- 200
- 300
- 400
- 500
- 600

Number of SPEs
- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8

Speedup
- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
FFT on Cell

• Williams et al. analyzed peak performance of FFT of various types.
• Green and Cooper showed impressive results for an FFT of size 64K.
• Chow et al. developed a design for 16 million complex samples.
• FFTW supports FFT of various size, type and accuracy.
• None exhibit good performance for small input size
FFT Algorithm used: Cooley Tukey

Algorithm 1: Sequential FFT algorithm

\[
\begin{align*}
\text{Input: array } & A[0] \text{ of size } N \\
1 & \text{NP } \leftarrow 1 \\
2 & \text{problemSize } \leftarrow N \\
3 & \text{dist } \leftarrow 1 \\
4 & i1 \leftarrow 0 \\
5 & i2 \leftarrow 1 \\
6 & \text{while problemSize } > 1 \text{ do} \\
7 & \hspace{1em} \text{Begin Stage;} \\
8 & \hspace{1em} a \leftarrow i1 \\
9 & \hspace{1em} b \leftarrow i2 \\
10 & \hspace{1em} k = 0, j\text{twiddle } = 0 \\
11 & \hspace{1em} \text{for } j \leftarrow 0 \text{ to } N - 1 \text{ step } 2 \ast \text{NP do} \\
12 & \hspace{2em} W \leftarrow w(j\text{twiddle}) \\
13 & \hspace{2em} \text{for } j\text{first } \leftarrow 0 \text{ to } \text{NP do} \\
14 & \hspace{3em} b[j + j\text{first}] \leftarrow a[k + j\text{first}] + a[k + j\text{first} + N/2] \\
15 & \hspace{3em} b[j + j\text{first} + \text{Dist}] \leftarrow (a[k + j\text{first}] - a[k + j\text{first} + N/2]) \ast W \\
16 & \hspace{2em} k \leftarrow k + \text{NP} \\
17 & \hspace{2em} j\text{twiddle } \leftarrow j\text{twiddle} + \text{NP} \\
18 & \hspace{1em} \text{swap}(i1, i2); \\
19 & \hspace{1em} \text{NP } \leftarrow \text{NP } \ast 2 \\
20 & \hspace{1em} \text{problemSize } \leftarrow \text{problemSize}/2 \\
21 & \hspace{1em} \text{dist } \leftarrow \text{dist } \ast 2 \\
22 & \hspace{1em} \text{End Stage;}
\end{align*}
\]

\text{Output: array } A[i1] \text{ of size } N

- Out of Place 1D FFT requires two arrays A & B for computation at each stage
- Saves bit-reversal stage

Georgia Tech
College of Computing

Virat Agarwal, STI Cell workshop
FFT Algorithm for the Cell/B.E.

**Parallelization**
- Number of chunks = $2^p$, where $p$ : Number of SPEs
- Chunk $i$ and $i+p$ are allocated to SPE $i$
- Each chunk is fetched using DMA get with multibuffering

**Tree Synchronization**
- Synchronization after every stage using Inter-SPE DMA communication, Achieved in $(2\log n)$ stages.
- Each synchronization stage takes 1 microsec
  - PPU-coordinated synchronization takes 20 microsec.
FFT: Optimization for SPE

- Loop duplication for Stages 1 & 2
  - For vectorization of these stages we need to use spu_shuffle on output vector.

- Loop duplication for NP<buffersize and otherwise.
  - Need to stall for DMA get at different places within the loop.

- Code size increases which limits the size of FFT that can be computed.
FFT: Design Challenges

- Synchronization step after every stage (log N) leads to significant overhead
  - minimize the sync. time by using tree based approach using inter SPE comm.

- Limited local store
  - require space for twiddle factors and input data.
  - loop unrolling and duplication increases size of the code.

- Algorithm is memory-bound
  - use multi-buffering - further increases the required space in a limited local store.

- Code is branchy with a doubly nested for loop within the outer while loop, lack of branch predictor compromises performance.
FFT: Performance analysis

Performance Comparison of our optimized FFT implementation as compared with other architectures.

Operation Count: \((5*N \log N)\) floating point operations
FFT: Performance Analysis

Pipeline utilization

Analysis of Pipeline utilization using asmviz tool from IBM.

Have a few stalls that still need to be optimized.
RC5: Encryption/Decryption on the Cell/B.E.

- Symmetric block cipher
  - Used in message encryption, digital signatures, stream encryption, internet e-commerce, file encryption, electronic cash.

- Three parameters
  - Word size $w$: 16, 32 or 64. (each block has 2 words)
  - Number of rounds $r$: 0,1,2,…,255
  - Number of bytes in key $b$ :0,1,2,..,255

- Mixing the secret key
  - Encryption: Left rotate operation encrypts the input
  - Decryption: Right rotate operation is used to generate the output.
**RC5: Optimizing for the Cell**

- Divide input array into $p$ equal chunks, where $p$ is the number of SPEs. Each SPE ($i$) is allocated chunk $i$.

- The chunks are fetched into the SPEs using double-buffering.
  - The data indices in the fetched buffer are shuffled so as to form two separate chunks of input data. These two separate arrays are then used as the 2 blocks for RC5 encryption.

- The *for* loop in both RC5 encryption/decryption is vectorized and unrolled for best pipeline utilization.

- RC5 assumes a little-endian input, for Cell we need to convert to the required convention, which compromises performance.
RC5: Performance Analysis

Running Time of RC5 encryption with varying number of SPEs

Running Time of RC5 decryption with varying number of SPEs

RC5 encryption on Cell PPE-only vs Cell optimized

RC5 decryption on Cell PPE-only vs Cell optimized

Virat Agarwal, STI Cell workshop
MPEG-2 decoding : Parallel Algorithm

• SMP algorithm [Bilas, Fritts, & Singh, 1996]
  – Authors examined various points of parallelization (Groups of Pictures vs. Slices)
  – Determined that parallelizing on Slices is most efficient when running on multiple processors (low bandwidth communication, high amount of local storage)

1. Add pictures to the picture task queue as they are encountered in the bitstream
2. For each picture, decode the slices and add them to a task queue that worker processors take work from
3. Decode the slices in parallel, synchronizing at the end of each picture
4. When all slices of a picture are decoded, remove it from the picture task queue and add it to the display queue

• Design Considerations
  – Since a slice can contain an arbitrary number of macroblocks, workload imbalance can occur (usually not an issue since the number of slices per picture is limited)
  – Can significantly improve performance by synchronizing at the end of specific pictures
  – Other points of parallelization are dismissed by the authors due to load imbalance
MPEG-2 Decoding on Cell

- We parallelize on macroblocks
  - The SMP algorithm authors dismissed other points of parallelization since they created workload imbalance, assuming all workers can perform all tasks at equal efficiency
  - For macroblock parallelization, all workers must wait while one processor completely decodes a picture

  1. One PPE thread decodes each picture and adds all macroblocks to a work queue
  2. Another PPE thread assigns work to the SPEs from the work queue
  3. Each SPE is used to perform Inverse Discrete Cosine Transform and scatter/gather operations (computationally expensive operations)

- Design Considerations
  - SPE performs scatter/gather operations on small amounts of memory, which is inefficient
  - Workload is divided unevenly between the PPE and SPEs (SPEs are used exclusively as accelerators)

- Initial performance results
Acknowledgment of Support

• National Science Foundation
  - CSR: A Framework for Optimizing Scientific Applications (06-14915)
  - CAREER: High-Performance Algorithms for Scientific Applications (06-11589; 00-93039)
  - ITR: Building the Tree of Life – A National Resource for Phyloinformatics and Computational Phylogenetics (EF/BIO 03-31654)
  - ITR/AP: Reconstructing Complex Evolutionary Histories (01-21377)
  - DEB Comparative Chloroplast Genomics: Integrating Computational Methods, Molecular Evolution, and Phylogeny (01-20709)
  - ITR/AP(DEB): Computing Optimal Phylogenetic Trees under Genome Rearrangement Metrics (01-13095)
  - DBI: Acquisition of a High Performance Shared-Memory Computer for Computational Science and Engineering (04-20513).

• IBM PERCS / DARPA High Productivity Computing Systems (HPCS)
  - DARPA Contract NBCH30390004

• IBM Shared University Research (SUR) Grant

• Sony-Toshiba-IBM (STI)

• Microsoft Research

• Sun Academic Excellence Grant
Thank you

Questions?