Optimizing Large Scale Chemical Transport Models for Multicore Platforms

John Linford and Adrian Sandu
Computational Science Laboratory
Virginia Tech
Introduction: How do we apply new technology to existing algorithms? Pros? Cons?

• Modern hardware is parallel, becoming massively parallel, becoming heterogeneous
• Parallel hardware is increasingly diverse
  – Shared and distributed memory in the same system
  – Homo-/Heterogeneous clusters of Homo-/Heterogeneous hardware
• Air quality models are computationally-intense multi-scale multi-physics models
  – Computationally-intense $\rightarrow$ motivates parallelism
  – Multi-scale $\rightarrow$ leverages a wide range of algorithms
  – Multi-physics $\rightarrow$ leverages a wide range of technologies
Comprehensive AQMs motivate parallelization

Chemical kinetics

Transport & Meteorology

Aerosols

Emissions

STEM

108 Species

90 x 60 x 18 x 108 = 10,497,600 variables

(10,497,600 x 64) / 8 = ~80MB per iteration
Mass-balance equations for trace species determine the fate of pollutants in the atmosphere

\[
\frac{\partial \mathbf{c}}{\partial t} = -\mathbf{u} \cdot \nabla \mathbf{c} + \frac{1}{\rho} \nabla \cdot (\rho \mathbf{K} \nabla c_i) + \frac{1}{\rho} f_i(\rho c) + E_i
\]

1a. \( \mathbf{c}_i(t^0, x) = \mathbf{c}_i^0(x) \)

1b. \( \mathbf{c}_i(t, x) = \mathbf{c}_i(t, x), \quad x \in \Gamma^I \)

1c. \( \mathbf{K} \frac{\partial \mathbf{c}_i}{\partial n} = 0, \quad x \in \Gamma^O \)

1d. \( \mathbf{K} \frac{\partial \mathbf{c}_i}{\partial n} = V_i^{dep} \mathbf{c}_i - \mathbf{Q}_i, \quad x \in \Gamma^G, \text{ for all } 1 \leq i \leq s. \)

These equations are solved in a sequence of \( N \) time steps of length \( \Delta t \) taken between \( t^0 \) and \( t^N = T. \)  \cite{Sandu et al., 2005}
Mass-balance equations for trace species determine the fate of pollutants in the atmosphere

\[ \partial \Omega = \Gamma^I \cup \Gamma^o \cup \Gamma^G \]

1a. \[ \frac{\partial c_i}{\partial t} = -u \cdot \nabla c_i + \frac{1}{\rho} \nabla \cdot (\rho K \nabla c_i) + \frac{1}{\rho} f_i(\rho c) + E_i \]

1b. \[ c_i(t^0, x) = c_i^0(x) \]

1c. \[ c_i(t, x) = c_i^I(t, x), \ x \in \Gamma^I \]

1d. \[ K \frac{\partial c_i}{\partial n} = 0, \ x \in \Gamma^O \]

1e. \[ K \frac{\partial c_i}{\partial n} = V_i^\text{dep} c_i - Q_i, \ x \in \Gamma^G, \text{ for all } 1 \leq i \leq s. \]

These equations are solved in a sequence of \( N \) time steps of length \( \Delta t \) taken between \( t^0 \) and \( t^N = T \). \[ \text{[Sandu et al., 2005]} \]
Mass-balance equations for trace species determine the fate of pollutants in the atmosphere

Boundary

\[ \partial \Omega = \Gamma^I \cup \Gamma^O \cup \Gamma^G \]

1a. \[ \frac{\partial c_i}{\partial t} = -u \cdot \nabla c_i + \frac{1}{\rho} \nabla \cdot (\rho K \nabla c_i) + \frac{1}{\rho} f_i(\rho c) + E_i \]

1b. \[ c_i(t^0, x) = c_i^0(x) \]

1c. \[ c_i(t, x) = c_i^I(t, x), \ x \in \Gamma^I \]

1d. \[ K \frac{\partial c_i}{\partial n} = 0, \ x \in \Gamma^O \]

1e. \[ K \frac{\partial c_i}{\partial n} = V_i^{dep} c_i - Q_i, \ x \in \Gamma^G, \ \text{for all} \ 1 \leq i \leq s. \]

These equations are solved in a sequence of \( N \) time steps of length \( \Delta t \) taken between \( t^0 \) and \( t^N = T \). [Sandu et al., 2005]
Mass-balance equations for trace species determine the fate of pollutants in the atmosphere

Boundary
\[ \partial \Omega = \Gamma^I \cup \Gamma^O \cup \Gamma^G \]

1a. \[ \frac{\partial c_i}{\partial t} = -u \cdot \nabla c_i + \frac{1}{\rho} \nabla \cdot (\rho K \nabla c_i) + \frac{1}{\rho} f_i(\rho c) + E_i \]
1b. \[ c_i(t^0, x) = c_i^0(x) \]
1c. \[ c_i(t, x) = c_i^I(t, x), \quad x \in \Gamma^I \]
1d. \[ K \frac{\partial c_i}{\partial n} = 0, \quad x \in \Gamma^O \]
1e. \[ K \frac{\partial c_i}{\partial n} = V_i^{dep} c_i - Q_i, \quad x \in \Gamma^G, \text{ for all } 1 \leq i \leq s. \]

These equations are solved in a sequence of \( N \) time steps of length \( \Delta t \) taken between \( t^0 \) and \( t^N = T. \)  

[Sandu et al., 2005]
FIXEDGRID is a prototype AQM for multicore Systems

- Simplified multi-scale atmospheric model
- Explicit time-stepping in transportation
- SAPRC’99 chemical mechanism
  - 79 species, 211 reactions, stiff system
- Research code to easily compare a range of computing platforms
- Written in Fortran 90
FIXEDGRID uses dimension splitting to simplify computation and reduce round-off error

- X / Y dimension splitting reduces program complexity
- Rows / Columns explicitly discretized and solved with the same code routine
- Symmetric 2\textsuperscript{nd} order time-dimension splitting reduces round-off error
The Cell Broadband Engine is a heterogeneous CMP

- **1 Power Processor Element**
  - 64-bit, dual threaded
  - Vector/SIMD extensions
- **8 Synergistic Processor Elements**
  - 128-bit, single threaded
  - Vector/SIMD extensions
  - 128 128-bit registers
  - 256KB local storage
  - Optimized for 32-bit single-precision floating point
  - Memory Flow Controller supplies asynchronous DMA
- **204.8GB Peak EIB Bandwidth**
The Cell memory model has “interesting features”

- SPEs must use DMA to access main memory
  - Data must be “correctly” aligned
  - Minimum transfer size is 16 bytes
  - Data must be contiguous
  - Data should be 128-byte aligned

- SPEs have only 256K of local storage (~8,000 doubles)
- PS3 reserves two SPEs
- PS3 has less memory than CBE blade
We produced four versions of FIXEDGRID

1. Translated Fortran to C
2. Moved computational cores to the SPEs
3. Optimized SPE code
4. Implemented scalable strided main-memory access method

<table>
<thead>
<tr>
<th></th>
<th>Ver. 1</th>
<th>Ver. 2</th>
<th>Ver. 3</th>
<th>Ver. 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2638</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>3965.45</td>
<td>1271.25</td>
<td>1118.95</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>2095.02</td>
<td>771.30</td>
<td>632.94</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>1430.63</td>
<td>678.99</td>
<td>490.58</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>1137.48</td>
<td>658.42</td>
<td>423.19</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>988.32</td>
<td>643.89</td>
<td>389.89</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>928.20</td>
<td>646.30</td>
<td>375.96</td>
</tr>
</tbody>
</table>

Runtime in seconds for 1000 x 700 grid points
FIXEDGRID Version 1: Baseline

• No Fortran compiler in CBE SDK 2.2
  – SDK 3.0 includes a modified version of XL Fortran
• Automatic Fortran $\rightarrow$ C translation is available, (i.e. f2c) but we did it by hand
  – Improve source code readability
  – Refactor code to better fit the execution model
• Translation “rules of thumb:”
  – parameter $\rightarrow$ #define
  – using module $\rightarrow$ #include
  – Fortran array syntax $\rightarrow$ C macros (when possible)
• Long, tedious, bug-prone, not recommended for real-world codes (approx. 38 work hours, $\sim$2,000 lines)
FIXEDGRID Version 2: Parallelized baseline

- Offloaded the main computational cores: `discretize()` and `advec_diff()`
- Implemented generic data types for passing arguments to the SPUs
- Implemented tiny 32-bit communication library for PPU / SPU communication
  - Send/receive to/from SPUs
  - Broadcast to SPUs
  - Synchronize PPU with one/all SPUs
SPEs copy matrix rows directly from main memory
PPU buffers transposed columns in main memory
Reordering matrix column data is a severe bottleneck

Column discretization bounded by array copy

Scalability limited by memory bottleneck
FIXEDGRID Version 3: Optimized SPU code

<table>
<thead>
<tr>
<th>Optimization</th>
<th>Runtime Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double-buffered DMA</td>
<td>23%</td>
</tr>
<tr>
<td>Vectorization and SPU intrinsics</td>
<td>18%</td>
</tr>
<tr>
<td>Loop unrolling and branch prediction</td>
<td>8%</td>
</tr>
</tbody>
</table>

Mem[0, BSIZE) → Buffer[0]
Mem [BSIZE, 2*BSIZE) → Buffer[1]

for i in 0, 1, 2, ... MATRIX_SIZE / BSIZE
  b = i mod 2

  Process Buffer[b]
  Buffer[b] → Mem[i*BSIZE, (i+1)*BSIZE]
  Mem[(i+2)*BSIZE, (i+3)*BSIZE) → Buffer[b]
end

vector double sum_d;
vector double vec1_d =
  {buff[i], buff[i+1]};
vector double vec2_d =
  {buff[i+2], buff[i+3]};
sum_d = spu_add(vec1_d, vec2_d);

Fetch, process and write any remaining buffer
FIXEDGRID Version 4: Scalable non-contiguous DMA

• Use DMA lists to reduce overhead for multiple consecutive transfers
• Minimum transfer size is 16 bytes, so transfer two double precision variables
• Columns arrived interleaved into SPE local storage
Scalable strided DMA requires $O(n)$ overhead storage

- DMA lists reside in SPE local storage
- List of pairs of 64-bit address lower bits and transfer lengths
- One DMA list element is required for each 16 byte transfer

```c
typedef struct {
    union {
        unsigned int all 32;
        struct {
            unsigned nbytes: 31;
            unsigned stall: 1;
        } bits;
    } size;
    unsigned int ea_low;
} dma_list_elem_t;
```
FIXEDGRID performs exceptionally well on the Cell
We parallelized FIXEDGRID Version 1 with OpenMP

- Trivial to implement, less than 4 hours, ~50 lines of code
- Test platform: dual Intel Quad-Core Xeon shared-memory workstation
FIXEDGRID scales well on a homogeneous system.
Conclusions

• Heterogeneous accelerated CMPs can significantly improve the performance of large-scale chemical transport models
• Porting a small Fortran code to CBE was non-trivial. Porting larger codes would be impractically difficult without better tools.
• Hardware support for non-contiguous memory access in CBE should be considered.
• Alternate matrix storage formats should be considered for accelerated CMPs with limited main-memory access from accelerators
Thank you