Title

# Impacts of the IBM Cell Processor to Support Climate Models



Authors

Zhou, Shujia

Duffy, Daniel

Clune, Tom

et al.

Publication Date

2008-06-18

DOI

10.2172/1407063

Peer reviewed

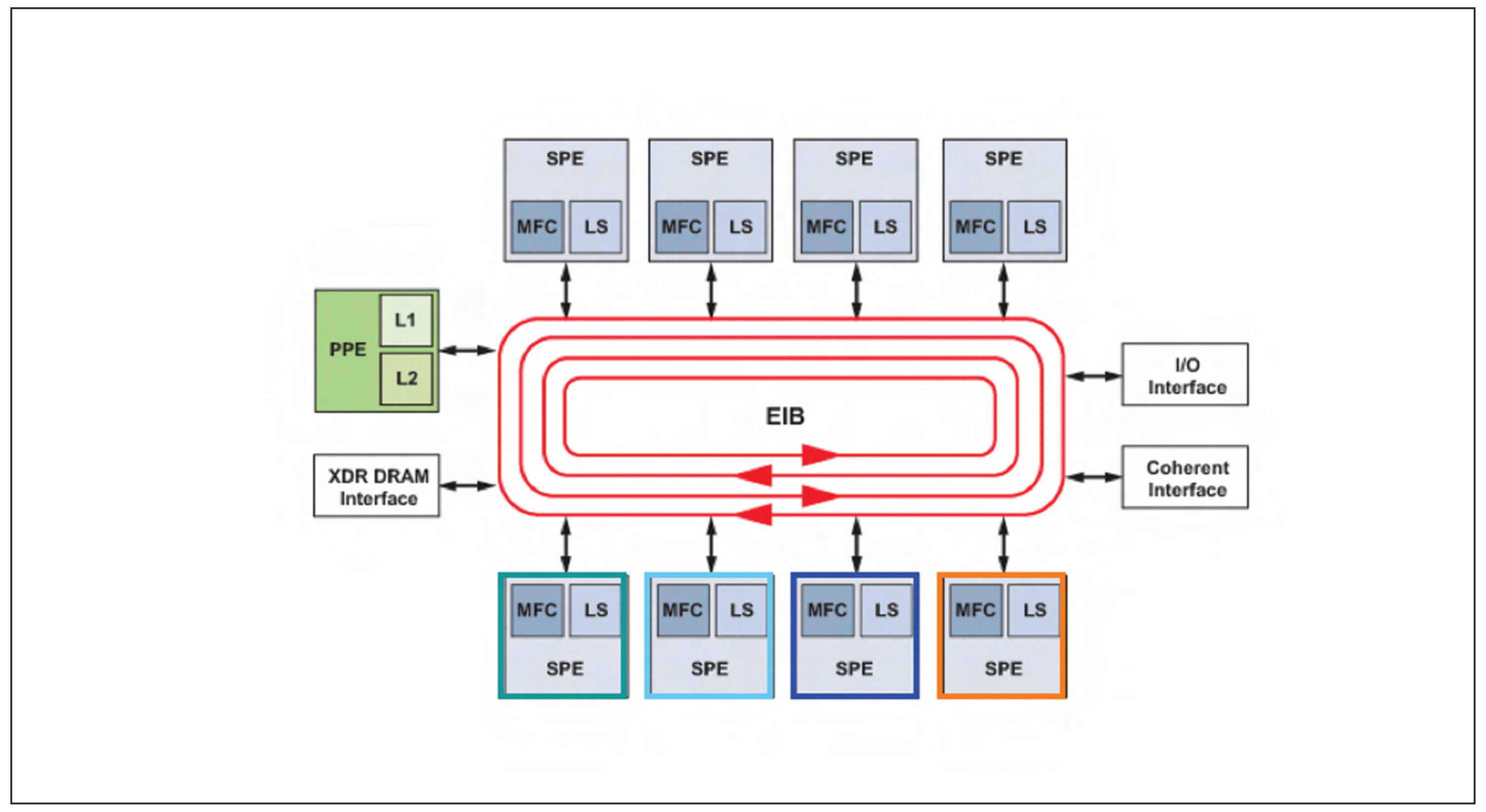
# Impacts of the IBM Cell Processor on Supporting Climate Models

Shujia Zhou[1,2], Daniel Duffy[1,3], Tom Clune[1], Max Suarez[1], Samuel Williams[4], Milt Halem[5]

## Introduction:

NASA is interested in the potential performance and cost benefits of adapting some science applications to emerging nontraditional processors such as the IBM Cell Broadband Engine System (hereafter referred to as "Cell"). The Cell is a multi-core system with the capability of increasing performance by one to two orders of magnitude over traditional processors. However, the Cell's characteristics, 256K byte local memory in a single SPE (Synergistic Processing Element) as well as the new low-level communication mechanism, make it very challenging to port a full application such as the NASA Goddard Earth Observing System Model, Version 5 (GEOS-5). Like other climate and weather models, GEOS-5 consists of dynamics and column physics. To avoid the complexity of porting a full application in this feasibility study, we selected a single component of GEOS-5, namely the solar radiation component, which has been used in various climate and weather models.

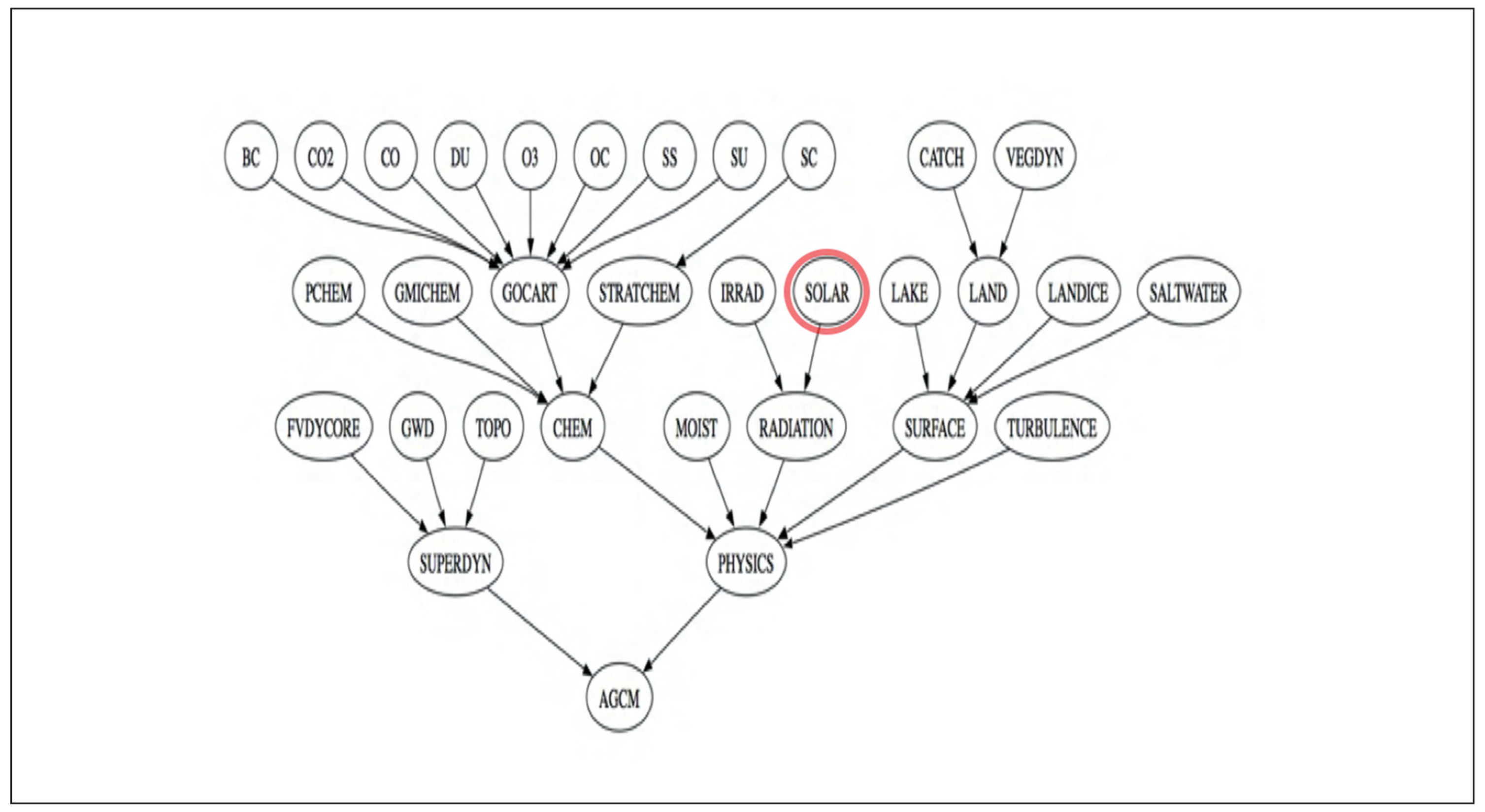
## CELL Chip:




- 205 single-precision GFLOPS
- High-speed data ring (EIB) with a sustained bandwidth of 205 GB/s
- 25.6 GB/s processor-to-memory bandwidth
- 256 KB local store at SPE

## NASA GEOS-5 Atmospheric Model:

The GEOS-5 atmospheric model consists of a large number of interacting physical components. Although not large in size (~2000 line Fortran code), the solar radiation component, the ultimate source of energy for the Earth's climate, is one of the most computationally-intensive components. Along with the infrared radiation component, it consumes at least 20% of the computing time of the atmospheric model. The requirements of the solar radiation component in loading and storing data from and to main memory are also small relative to the computational load. These factors make the solar radiation component well-suited for one of Cell's characteristics: a high ratio of computation to data transfer capabilities. In addition, this component consists of calculations in each column (along the vertical direction, on a grid point), which are independent of other columns, so-called "embarrassingly parallel." This greatly simplified the coding involving the Cell communication mechanism.

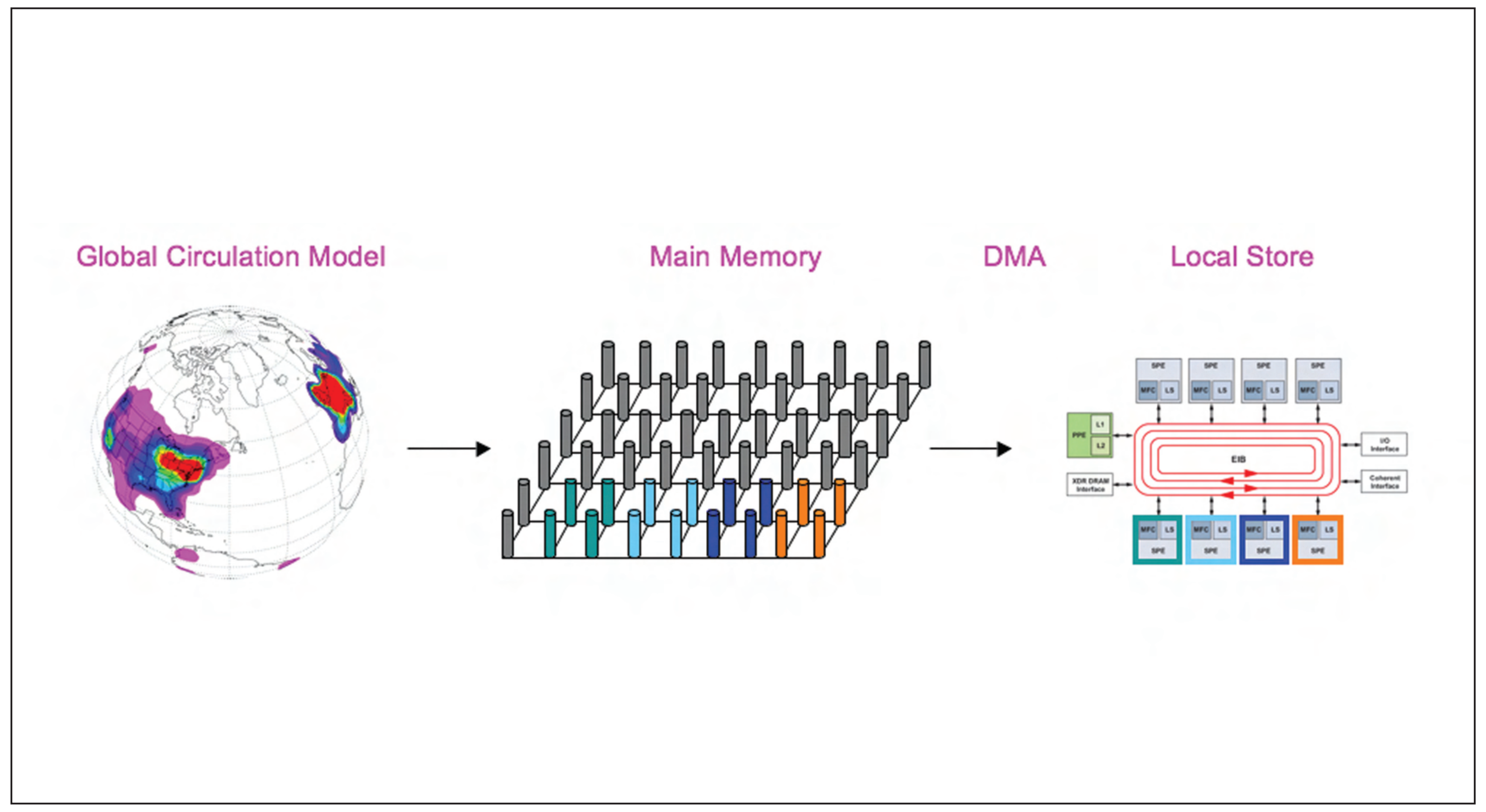



## Translate the Code from Fortran to C and Port It to Cell:

We converted the ~2000-line solar radiation code from Fortran to C (due to a Fortran compiler being unavailable for the SPE as of summer 2007). Converting the code includes three steps: (1) translating the code from Fortran to C, (2) inserting library calls to transfer data between the PPE and SPE with DMA (Direct Memory Access), and (3) vectorizing the most computationally-intensive function (currently a manual process not performed by the compiler). We found that 16-byte alignment and direct management of memory address (mapping the local multi-dimensional array index in SPEs to the global array index in main memory) required considerable time and attention to implement, since there are 27 various shapes of 1D, 2D, and 3D arrays involving data transfer between PPE and SPEs through DMA. However, we believe that conceptually these modifications are not difficult for a user who knows C and MPI. Steps (1) and (3) took ~3 weeks each. However, the newly released IBM XL V11.1 Fortran compiler should make steps (1) and (3) unnecessary. At this time, we do not have access to this Fortran compiler and therefore have not evaluated its performance, particularly regarding autovectorization. Hence, step (3) might still be necessary for further performance improvement. Step (2) is unaffected by the availability of foreseeable Fortran compilers. However, a Fortran compiler will require writing a wrapper to use DMA, since the DMA's API is in C.

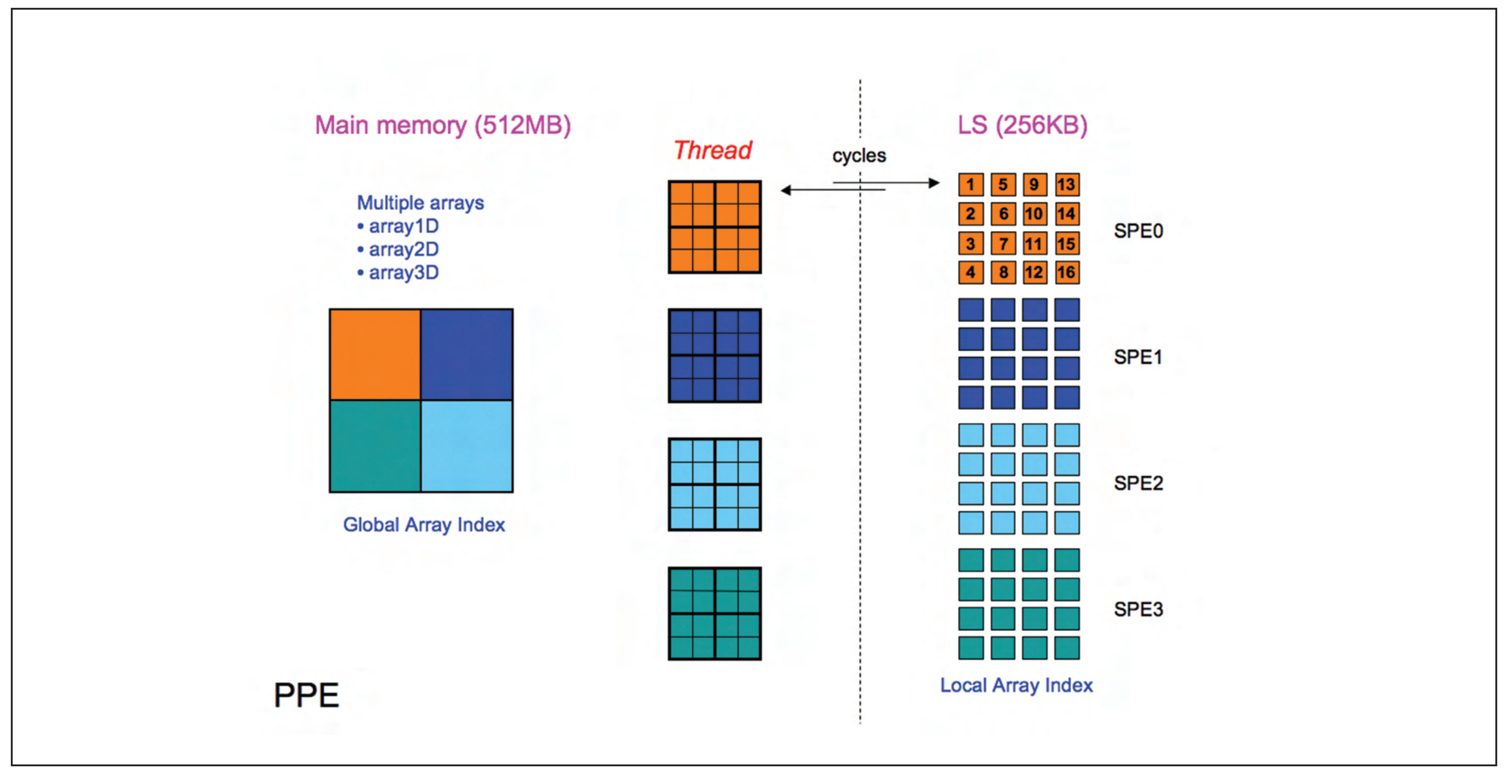
## Architecture:

Decomposing the solar radiation code for SPEs



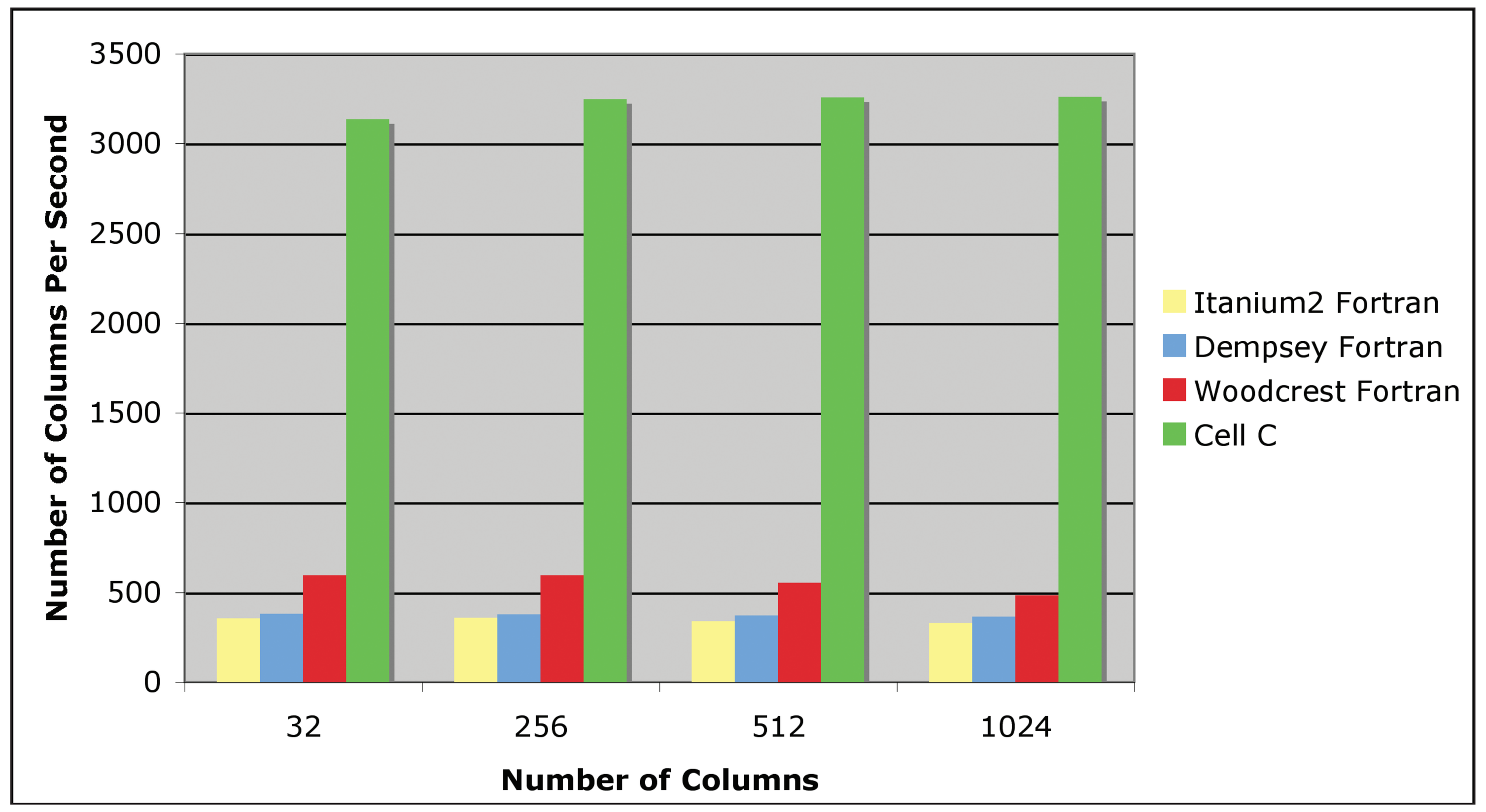

Processing four columns, each cycle using one SPE, is the optimal case

## Flow Diagram of Data Transfer via DMA (4 SPEs):




## Performance Evaluation of C-Version Cell Code vs. Original Baseline in Fortran:

- Since the current Cell processor in an IBM BladeCenter QS20 has a deficiency in processing double-precision floating point operations, we will only present results for the single-precision C-version solar radiation code. Our detailed memory analysis reveals that each SPE can contain four entire columns of data in local memory simultaneously, allowing us to further optimize the performance by vectorizing (SIMDizing) two computationally-intensive functions across these four independent columns. In comparison with the best baseline results (the single-precision, Fortran-version code), we found that in the case of 1024 columns per Intel core, a Cell with eight SPEs is 6.76x faster than Intel Xeon Woodcrest (2.66GHz, four floating point operations per cycle), 8.91x faster than Intel Xeon Dempsey (3.2 GHz, two floating point operations per cycle), and 9.85x faster than Intel Itanium2 (1.5 GHz, four floating point operations per cycle), respectively.
- We note that the C-version runs measurably slower than the Fortran baseline by factors of 1.46x, 2.56x, and 5.81x on Woodcrest, Dempsey, and Itanium2, respectively. Here we assume that the Cell performance is not hampered by our language conversion, and therefore will base our performance comparisons against the Fortran baseline. (Note: the O3 option is used for the compilation on all the processors reported here.)
- Beyond the Cell's powerful computational capability, we believe there are two essential factors for dramatic performance improvement over cache-based Intel processors: (1) fitting all computation-support data such as look-up tables into the SPE's local store, and (2) explicitly transferring data between SPEs and main memory via DMA. Once the data is in the local store, there are no effects from cache and TLB (Translation Lookaside Buffer), and DMA is a small fraction of total time.
- The more than 6x improvement on the solar radiation component is certainly very encouraging since the other column physics components such as infrared radiation and moisture have a similar code structure. That means ~50% of the total computational load for the model can expect to obtain significant performance benefits on the Cell.




## Summary:

- We found that the IBM Cell technology clearly provides a new way of dramatically improving the performance of climate and weather applications.
- Identifying the appropriate algorithms is the key to lowering the porting cost. In particular, we found that NASA GEOS-5 column physics components can improve their performance with Cell technology.

## Acknowledgments:

We would like to thank Carlos Cruz and Bruce Van Aartsen for translating some of the code from Fortran to C. Further, we would like to thank Tsengdar Lee (NASA High End Computing Program) for providing funding, Phil Webster for project initiation, Mike Seablom for his inspiration and helpful discussion, John Shalf for sharing his insight on the IBM Cell technology, the NASA Center for Computational Sciences for installing the IBM Cell Simulator for code development, the Dice Project for training support, and the UMBC Multicore Computational Center for providing access to an IBM BladeCenter QS20.

[1] NASA Goddard Space Flight Center, [2] Northrop Grumman Corporation, [3] Computer Sciences Corporation, [4] University of California, Berkeley, [5] University of Maryland, Baltimore County