The switch to Multi-core has elevated concurrency as a major issue in utilizing the ever increasing number of processors on a single chip. This issue is further complicated in heterogeneous multi-cores, because different types of resources need to be individually optimized in order to achieve maximum global performance.

The Cell processor is a state-of-the-art high performance heterogeneous multi-core. It has one general purpose processor called the PPE, and eight SIMD processors, called the SPEs. Each SPE can directly operate on a small local store memory and can also access a common shared main memory through DMA calls. The memory management between the two levels is handled explicitly by software. Cell provides a high computational power on a single chip, making it a very appealing target for high-performance applications. However, because of its heterogeneity and its novel architectural elements, programming it is not a trivial process.

The Data-Driven Multithreading Virtual Machine (DDM-VM) addresses this challenge by adopting the Data Driven Multithreading (DDM) model of execution for the Cell processor. DDM is a non-blocking multithreading model that is based on the Data-Flow model of execution. The DDM model combines the latency tolerance and the distributed concurrency mechanisms of the Data-Flow model with the efficient execution of the sequential model.

DDM-VM implements the Thread Scheduling Unit (TSU) responsible for scheduling threads dynamically at runtime, as a software module running on the PPE core, leaving the SPE cores to execute the threads. DDM-VM utilizes the DDM CacheFlow policy to handle data management and prefetching without the programmer intervention. Two configurations for CacheFlow are implemented and evaluated.

The programmer uses a set of compiler directives to describe the parallel sections of the code and the data they produce and consume. The program is then parsed with a Preprocessor Tool and the resulting code is compiled using the normal tool chain of the Cell and linked with the DDM-VM runtime library. The runtime implements all the tasks of scheduling, execution instantiation, synchronization, and data movement implicitly. The scheduling & data management tasks are overlapped with the execution of the threads allowing the system to tolerate memory and synchronization latencies.

The contribution of this work is an efficient virtual machine that uses Data-Flow synchronization techniques on a high performance heterogeneous multi-core system such as the Cell. Furthermore, DDM-VM uses the concepts of DDM and CacheFlow to automate the management of the two memory hierarchies of the Cell processor.

We have developed a prototype of the DDM-VM on a Sony Playstation 3 machine. For evaluation we have used a benchmark suit consisting of ten applications featuring kernels widely used in scientific and image processing applications.

The evaluation of two CacheFlow implementations showed that the distributed CacheFlow configuration, which distributes the DMA management to the part of the runtime running on the SPEs, is more efficient as it reduces the pressure on the PPE core running the TSU, which allows the system to scale well for a higher number of cores. For applications that benefit from data locality, namely Matrix Multiplication, the evaluation showed that CacheFlow can transparently exploit data locality and improve the performance. The overall performance evaluation on the benchmark applications showed that the platform generally scales well and tolerates synchronization and scheduling overheads efficiently. Furthermore, comparing the execution of three computationally intensive applications from our benchmark suit with CellSs and Sequoia, two other platforms that target the Cell, DDM-VM achieves better performance. We find these results very encouraging as they demonstrate the potential of models like DDM, that combine Data-Flow concurrency with efficient sequential thread execution, to be adopted as the execution model for Multi-Core systems.

We are currently working on porting the DDM-VM to a cluster of Cell Processors. We are also developing a GCC based source-to-source compiler tool for making the platform easier to program.

* This work was supervised by Professor Paraskevas Evripidou, University of Cyprus