CISC 879 Software Support for Multicore Architectures **Spring 2007**

Lecture 3: February 19

Lecturer: John Cavazos *Scribe: Varun N B, Sameer Kulkarni*

The following lecture gave the general overview of the various laws and the history behind the development of multicore architectures. The power point presentation for the lecture can be found at http://www.cis.udel.edu/ cavazos/cisc879/Lecture-03.ppt .

The two law discussed were :-

- Amdahla Law
- Gustafson's Law

3.1 Amdhal's and Gustafsons Laws

Lemma 3.1 *Amdalhls law states that the overall speed up that can be achieved by any process is limited by the weakest link in the process. With respect to parallelization the serial portions of the code form the weakest link.*

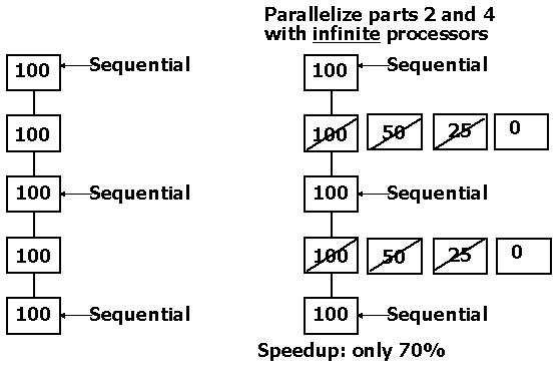

Figure 3.1: Amdhal's explanation

In postulating this theorem, Amdahl assumed work load to be fixed even when there is hardware / resource available for more parallelism.

Lemma 3.2 *Gustafsons law states that increasing processors gives linear speed up. More processors allow larger dataset size.*

CPU	GPU
Less Parallelism Simple Control requirements Devote more area for control and storage	More parallelism Complex control requirements More area for computational units
CPU Programming model -	
No data parallelism - no SIMD Small arithmetic units Less latency and less bandwidth	Exploit SIMD and provide for larger bandwidth for communication between processors

Table 3.1: Multi Core features

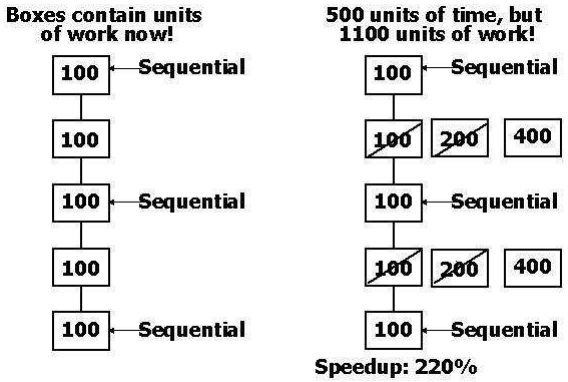

Figure 3.2: Gustafsons counter argument

Gustafson proposed fixed time and increased work load so that serial parts of the program have diminishing effect in reducing the overall speedup in a parallel environment. His tests revealed that, as processors grow the problem size is scaled and this scaling results in a substantial increase in the parallel parts of program as compared to the serial parts.

3.2 Different types of Multicore achitectures

3.2.1 Computational Power

Looking at the increase in the computational power we find that GPUs have increased their computational power at a much faster pace than CPUs. The GPU programming model differs from the general programming model in the following ways :-

- Streams - Collection of data sets that enable parallelization
- Kernel - Takes an input stream and delivers an output stream. No correlation between stream data.
- Stream storage - Input read once and output written once resulting in producer-consumer locality.

3.3 The Cell

The Cell is a heterogenous multicore architecture that is a hybrid of both GPU type and general Control type processors.

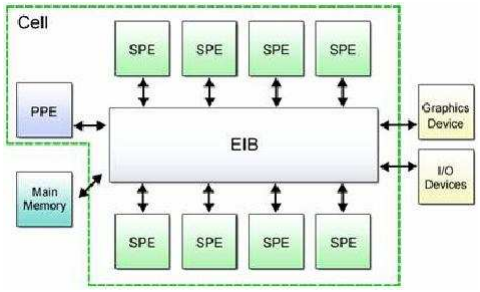

Figure 3.3: The Cell

The main features of the cell:-

1. Exploit parallelism
2. Achieve High Frequency
3. Higher Bandwidth through DMA
4. Applicable to variety of applications

The CELL heterogeneous architecture is composed of two main functional units.

3.3.1 SPE

- Synergistic processor element for data intensive processing. SPE features - No cache, large unified register, high band width interface bus for communication. Dedicated DMA engine.

3.3.2 PPE

- Power Processor Element handles the control tasks. General purpose 64 bit PowerPC RISC L1 and L2 cache Used for operating system and program control

GNU based C++ compilers and GDB debugger produce code and help debugging the two processors when invoked in different modes. Various performance profiling tools provided are OProfile, GProf, simulator and spu timing