Breaking the Sequential Dependency and Extracting Parallelism out of Recurrence Equations —Sample Application to Options Pricing in Finance and HMMER in Computational Biology

Narayan Ganesan, Roger D. Chamberlain and Jeremy Buhler

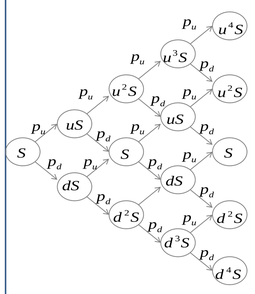
Department of Computer Science and Engineering, Washington University in St. Louis

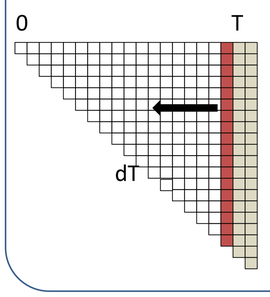
This research has been supported by NIH grant R42 HG003225 and Exegy, Inc. R.D. Chamberlain is a principal in Exegy.

Introduction

- Dynamic Programming problems and Uniform Recurrence Equations are ubiquitous.
- Many problems impose a sequential dependency among cells and are evaluated serially.
- The problem sizes are large and the number of problems is growing exponentially.
- Areas of applications:
  - Computational Biology – Sequence and Model Analysis(Smith-Waterman, Protein Motif Finding, RNA Folding etc.)
  - Finance – Lattice Methods for Options Pricing(Binomial and Trinomial Trees)
  - Numerical solutions to differential equations
  - Sorting etc.
- Need to efficiently parallelize the evaluation of recurrence equations(GPU/FPGAs/SIMD)
- Two sample applications from diverse fields are implemented using this novel method.

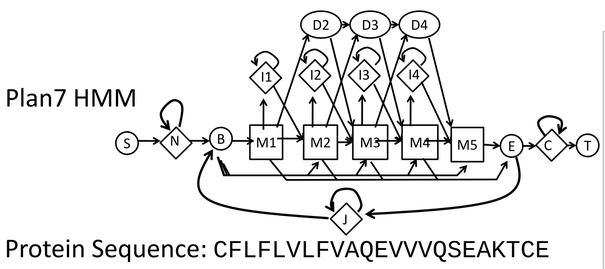
Application 1: Binomial Tree Options Pricing





- Nodes of the Binomial Tree are different asset price possibilities
- The nodes at the same depth correspond to single time instant
- The binomial tree can be represented by a Dynamic Programming matrix.
- The options price at the root node can then be evaluated starting from leaves and working back through time sequentially.
- A suitable parallel architecture would evaluate the cells corresponding to a single time-instant in parallel
- Serial Implementation - $\Theta(T^2)$ Parallel Implementation – $\Theta(T)$
- In the financial world a millisecond delay could mean significant loss, so the parallel implementation is targeted for speed-up.

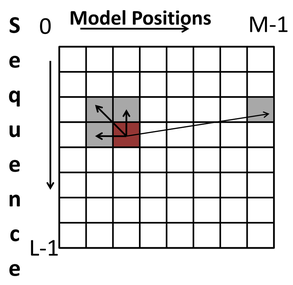
Application 2: Protein Motif Finding- HMMER search



Protein Sequence: CFLFLVLFVAQEVVVQSEAKTCE

Dynamic Programming Matrix:

- Each cell is dependent on three adjacent cells and the terminal cell from the previous row.
- This dependency imposes a serial restriction on their evaluation.



- Match a given protein sequence to a given HMM.
- HMM is described by the corresponding transition and emission probabilities.
- Viterbi Algorithm is used to evaluate the corresponding costs of matching.
- Gives rise to dependency on local cells of the dynamic programming matrix.
- Time Complexity – product of model and sequence lengths, $\Theta(|M|.|S|)$

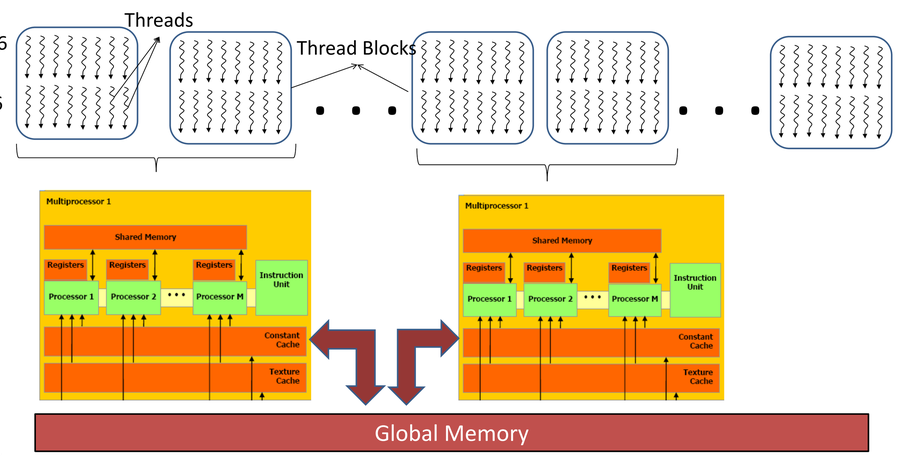
NVIDIA GPU – CUDA Programming Interface

Physical:

- 8 Scalar Processors within a MultiProcessor (MP) can execute 16 threads in parallel each.
- 128 way parallelism per MP or 256 to hide memory access latency.
- 10-100s of MPs in a GPU part.

Programming:

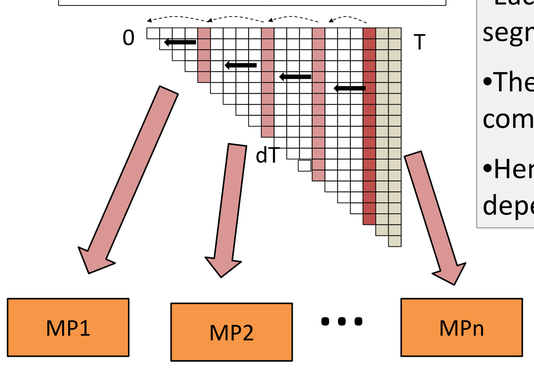
- Multiple thread blocks per MP
- Threads within a block can access fast shared memory.
- Threads across blocks can communicate via global memory.
- Data-Parallel applications can experience tremendous speed-ups.



Implementation: Parallelization and Acceleration

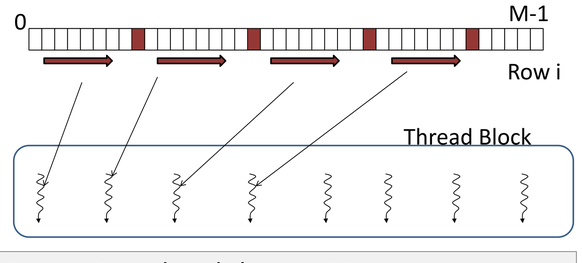
- The dependency between the cells can be broken and resolved later due to the associative nature of the operators in the recurrence equations.
- This property can be exploited to evaluate the cells independently and in parallel – even if the cells are sequentially dependent to begin with.
- Hence this approach converts data-dependent computation to data-parallel computation with minimum dependency, thus taking advantage of the parallel processing capability of GPUs.

Binomial Tree Options Pricing



- Each MP evaluates a consecutive segment of the array in parallel.
- The boundary values are communicated via global memory
- Hence the eventual speed-up would depend on communication latency.

HMMER Search



- Each Row is partitioned and the partitions are evaluated independently and in parallel.
- Threads of a single thread block are assigned to the partitions which communicate via fast shared memory.

Results and Conclusion

- The algorithm preserves locality of computation hence could experience speed-ups on other architectures as well(FPGA/GPU)
- Optimal speedup is given by, $\frac{\sqrt{N}}{2\sqrt{2L}}$ where N is number of sequentially dependent cells and L, the communication latency.

- **Binomial Tree** of depth 1024 was partitioned into 4 consecutive segments and evaluated on a GeForce 8600 with latency to global memory at ~500 clks in order to obtain a speedup of 2x over NVIDIA's parallel implementation.

- **HMMER Search** for a model of 507 positions and sequences of total length 65416. Baseline is single core of Intel Core2Duo@3GHz

GPU Model	No. of MPs	Speedup
Single Multiprocessor @504MHz	1	1.1x
GeForce 8600	4	4.3x
GTX260	24	27x
Tesla C1060	30	32x