

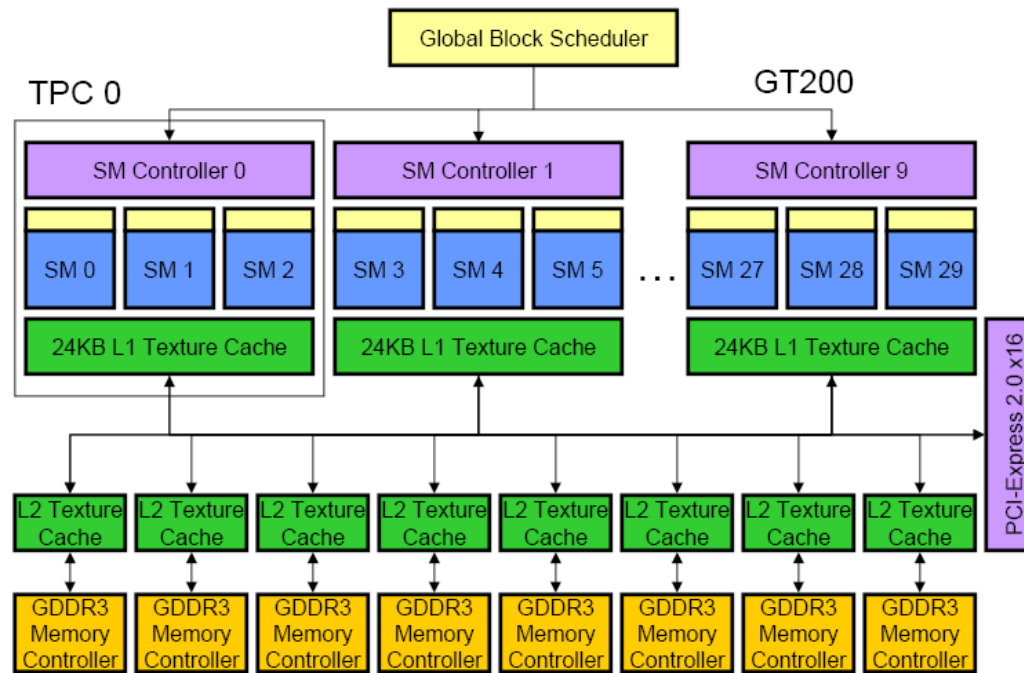
# Path Integral Option Pricing on NVIDIA GPUs

D. A. Dimitrov & I. Pogorelov  
Tech-X Corp., Boulder, CO

## Motivation

- The currently available NVIDIA Tesla S1070 server (\$8000) is a general-purpose parallel architecture with important computational processing power per \$.
- It consists of 4 Tesla 10 (T10) GPUs each with 4 GB RAM and peak performance of 1 TFLOPS single precision (IEEE 754 floating point) and 87 GFLOPS double precision.
- Our goal is to explore using this architecture for acceleration of a specific computational problem - pricing of American options.
- This architecture is particularly relevant to this problem, since a large number ( $\sim 100,000$ ) of option prices have to be computed per second.

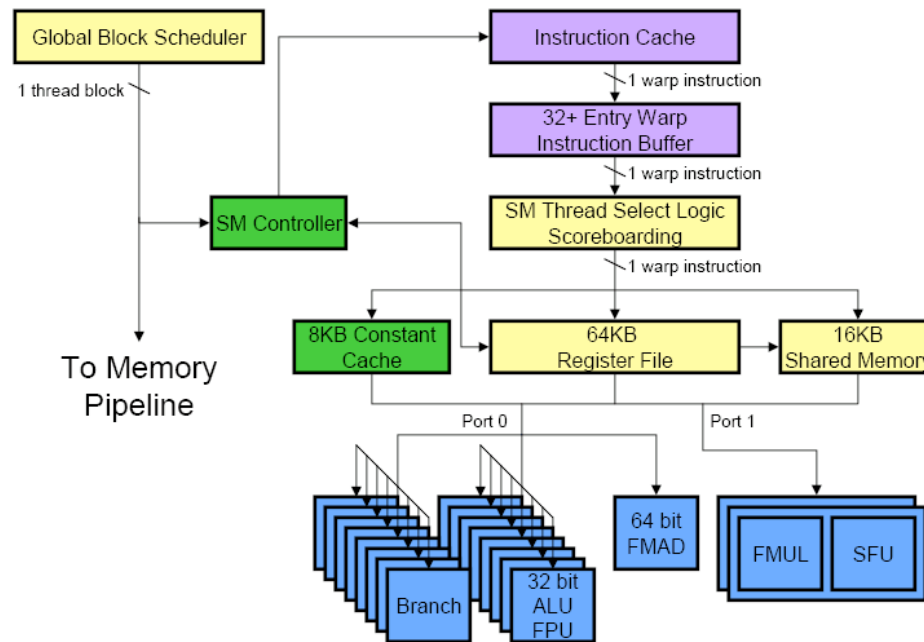
# NVIDIA GeForce GTX 200 GPU Architecture



From: "NVIDIA's GT200: Inside a Parallel Processor", D. Kantar (2008).

- The Tesla 10 GPU is assumed to be based on this architecture.
- A massively parallel architecture with 30 streaming multiprocessors (SM).

# GT200 Streaming Multiprocessor Architecture



From: "NVIDIA's GT200: Inside a Parallel Processor", D. Kantar (2008).

- 8 streaming processors, IEEE 754 32-bit floating point support.
- 1 Double Precision unit, IEEE 754 64-bit floating point support.
- 2 Special Function Units.

## Security Price Model

- Model security price evolution  $S_t$  via a geometric Brownian motion stochastic process:

$$dS_t = \mu S_t dt + \sigma S_t dW_t,$$

with  $W_t$  a Wiener process, constant percentage drift  $\mu$  and volatility  $\sigma$ .

- Given an initial value of  $S_0$ , the security price at time  $t > 0$  is a log-normally distributed random variable:

$$S_t = S_0 \exp \left( \left( \mu - \frac{\sigma^2}{2} \right) t + \sigma W_t \right).$$

## European Option Pricing

- Price of a put option with a strike  $K$  at maturity time  $T$ :

$$O_p(S_T) = \max(K - S_T, 0)$$

- Price at  $t < T$  if exercise allowed only at maturity (European Option) and the risk-free interest rate  $r$  is constant:

$$O(S_t) = e^{-r(T-t)} E [O(S_T) | S_t],$$

$E [\dots]$  is evaluated under the considered security price model with  $z_t = \ln(S_t)$ :

$$E [O(S_T) | S_t] = \int p(z_T | z_t) O(e^{z_T}) dz_T.$$

## American Option (AO) Pricing

- An American option is possible to exercise at any time up to and including the expiration date.
- For optimal pricing, the paths of the security price stochastic process have to be evaluated.
- We consider here two path integral based algorithms for pricing of American put options via evaluation at  $N_t$  number of exercise times up to and including the expiration date:

$$O(S_{t_{n-1}}) = \max \left\{ O^Y(S_{t_{n-1}}), e^{-r\Delta t} E [O(S_{t_n}) | S_{t_{n-1}}] \right\}, \quad (1)$$

where  $O^Y(S_{t_{n-1}})$  is the price in case of anticipated exercise at  $t_{n-1}$  and  $t_n = n\Delta t$ ,  $n = 1, \dots, N_t$ ,  $\Delta t = T/N_t$ .

## Path Integral Approach for AO Pricing

- Our implementation for AO pricing is based on the approach in:  
[1] G. Montagna *et al.*, *A Path Integral Way to Option Pricing*, Phys. A **310**, pp. 450-66 (2002) (arXiv:cond-mat/0202143v1).
- Both algorithms solve Eq. (1) backwards in time on specifically selected integration points for efficient option pricing.
- Algorithm 1 (ALG1) evaluates  $E [O (S_{t_n}) | S_{t_{n-1}}]$  via trapezoidal integration with  $N_z$  equispaced abscissas in  $z = \ln(S)$ .
- In Algorithm 2 (ALG2), these expectation values are calculated using a second order, in  $\sigma \sqrt{\Delta t}$ , approximate analytical expression with effective  $N_z = 3$ .

## Choice of Integration Points

- Integration points are chosen at conditional mean values for the security price at each time slice:

$$E [z_{t_n} | z_{t_{n-1}}] = z_{t_{n-1}} + \left( \mu - \frac{\sigma^2}{2} \right) \Delta t + k\sigma\sqrt{\Delta t},$$

with  $k = -m, \dots, m$ :  $N_z = 2m + 1$ .

- The number of points at a time slice  $n$  is:

$$n(N_z - 1) + 1.$$

- This is an efficient choice since it leads to a *linear* growth of integration points at which the price of a path-dependent option has to be evaluated.

## Complexity of the two AO Pricing Algorithms

- Space complexity:  $\mathcal{O}(N_t \times N_z)$ .
  - An array of size  $N_t \times (N_z - 1) + 1$  is sufficient to hold the evaluated option prices at the specified integration points in the CPU implementation.
  - For ALG2, the space complexity is effectively  $\mathcal{O}(N_t)$  since accurate enough option pricing can be obtained with  $N_z = 3$ .
- Time complexity:  $\mathcal{O}(N_t^2 \times N_z^2)$ .
  - For ALG2, the effective time complexity is  $\mathcal{O}(N_t^2)$  since the expectation value  $E[O(S_{t_n}) | S_{t_{n-1}}]$  is given by an analytical expression that can be evaluated using 3 multiply-add (MAD) operations.

## GPU Kernel Implementation of ALG1 - Main Part

```
...
i = numTemporalPoints - 1; numZi = i*(numSpatialPoints - 1) + 1;
T tf = expAT; T w, tmp;
__syncthreads();
// initialization done
while (i >= 0) {
    tf *= expMAdt;
    auxi = (numTemporalPoints - i)*numSpatialPointsOver2;
    for (int m = threadIdx.x; m < numZi; m += blockDim.x) {
        tp = REAL_ZERO;
        for (int n = 0; n < numSpatialPoints; ++n)
            tp += rp[m+n]*weight[n];
        w = (tmp = ( K - Si[auxi+m]*tf )) > 0. ? tmp : 0. ;
        wp[m] = tp > w ? tp : w;
    }
    --i; numZi = i*(numSpatialPoints - 1) + 1;
    SWAP(rp, wp);
    __syncthreads();
}
if (threadIdx.x == 0)
    d_OptionValue[optionIndex].optionValue = rp[0];
```

## AO Pricing in Excellent Agreement with other Numerical Methods

Table 1: Comparison of our implementation (on CPUs & GPUs) of the path integral (PI) ALG1 ( $N_t = 123$ ,  $N_z = 13$ ) and ALG2 ( $N_t = 160$ ,  $N_z = 3$ ) results on pricing American put options vs published values [1] ( $N_t = 200$ ,  $N_z = 13$ ) from finite difference (FD), binomial tree (BT), and Green Function (GF) numerical methods with  $t_0 = 0$ ,  $T = 0.5$ ,  $r = 0.1$ ,  $\sigma = 0.4$ , and  $K = 10$ .

$S_0$	FD	BT	GF	PI [1]	PI ALG1	PI ALG2
6	4.000	4.000	4.000	4.000	4.000	4.000
8	2.095	2.096	2.093	2.095	2.095	2.094
10	0.921	0.920	0.922	0.922	0.922	0.921
12	0.362	0.365	0.364	0.362	0.362	0.362
14	0.132	0.133	0.133	0.132	0.132	0.132

## GPUs Used for Development and Experiments

GPU	Shader Clock (GHz)	# Multi Processors	# Processor Cores	Memory Size (MB)	SP Peak Perf. (GFLOPS)	DP Peak Perf. (GFLOPS)
T10 (GTX 280)	1.296	30	240	4	933	78

- The CUDA Toolkit version 2.1 was used for the experiments on the T10 GPU.
- We also used (mainly for development) a GeForce 8600M GT GPU with 4 SMs (32 cores) at 0.75 GHz, 512 MB RAM with CUDA Toolkit 2.0. This GPU was connected to an Intel Core 2 Duo CPU at 2.5 GHz, 6 MB L2 cache, 2 GB RAM, running under Mac OS X 10.5.7.
- All CPU results are from runs on a single CPU core.

## Relative Errors

- Errors are estimated vs the CPU results in DP (since there is currently no analytical expression for American option put prices and the excellent agreement with published results).
- L1 relative errors are calculated over  $480 \leq N_O \leq 2000$  number of American put option evaluations and the maximum L1 relative error of an option evaluation.
- ALG1 American put options evaluated with  $N_t = 123, N_z = 13$ ; ALG2 are with  $N_t = 160, N_z = 3$ .
- ALG1, CPU SP - L1:  $2 \times 10^{-7}$ , max L1/Option:  $2 \times 10^{-6}$ .
- ALG1, GPU SP - L1:  $2 \times 10^{-6}$ , max L1/Option:  $7 \times 10^{-6}$ .
- ALG2, CPU SP - L1:  $8 \times 10^{-6}$ , max L1/Option:  $4 \times 10^{-5}$ .

## Acceleration of ALG1 on NVIDIA Tesla T10 GPU

- Measured vs a CPU in single (SP) and double (DP) precision.
- The CPU runs were done on a Quad-Core AMD Opteron Processor at 2.3 GHz with 512 KB of cache size per core, 8 GB total RAM, under a 2.6.18 64-bit SMP GNU/Linux kernel.
- The acceleration is determined from execution times for evaluating from 480 to 2000 American put options with randomly initialized security prices in  $\$6 \leq S_0 \leq \$14$ .
- The GPU runs were in SP only (produces sufficient accuracy) using from 32 to 384 threads per option.
- Acceleration:  $t_{CPU}^{SP}/t_{GPU}^{SP} = 152$  using 256 threads per option (over 100 times acceleration achievable with 128 threads),  $t_{CPU}^{DP}/t_{GPU}^{SP} = 34$  with 224 threads/option.

## Summary

- Our first implementation of the path integral based ALG1 for pricing of American options on an NVIDIA T10 GPU achieved 152 times acceleration vs the CPU implementation in single precision and 34 times vs the double precision.
- ALG2 shows the fastest execution time on the CPU in double precision: 0.22 ms per American option (AO) put price evaluation or 4545 AO/s.
- Even a factor of 10 acceleration of ALG2 on a Tesla S1070 with 4 T10 GPUs will provide over 180000 AO (theoretical) value prices per second.
- Future development: complete an efficient implementation of ALG2 on the GPU.