Parallel CMC Pricing Algorithm for High Dimensional American-Syle Options

Kai Chen
CSCI596_2011_Fall_Project

The Classification and Monte Carlo (CMC) Algorithm

• Traditional American-style option pricing procedures like finite differences and lattice-based methods:
  > very accurate for ≤3 underlying assets case
  > grow exponentially with the number of underlyings

• In high-dimension pricing problems, some Monte Carlo based algorithms have been proposed:
  > simulate sample paths for the price processes of the underlyings
  > use backwards recursion in a dynamic programming fashion to estimate the option price
  > LSM method of Longstaff and Schwartz uses least squares regressions to estimate the continuation value of the option for any given point in time
  > CMC Algorithm further developed the LSM method, but used a classification process to determine the exercise/hold boundary.


Motive: pricing American-style options with a large number of underlying assets is computationally intensive and usually requires several days of serial computational time on a single processor system.

Goal: identify where and how the computation could be split into independent parallel tasks and then apply the parallel programming to replicate the CMC algorithm.

Parallel Version of CMC Algorithm

Parallel CMC algorithm
1: for t=T to 0, backward recursion at each opportunity date t, do
2: for i=1 to N1, on each realized price vector p(t,i) from simulations, in parallel do
3: Computation of training points
4: end for
5: Classification using boosting
6: end for
7: for i=1 to N, on MC step after knowing the exercise boundary, in parallel do
8: The partial option price computation
9: end for
10: Estimation of the final option price by merging the partial prices.

Hybrid MPI/OpenMP Implementation

Experiment Setting: Pricing a Bermudan/American call option on the maximum of three assets using CMC Algorithm.

Data generation 60.49%
Monte Carlo simulation 36.40%
Boosting classification 1.11%

Time distribution of the sequential CMC algorithm

Hybrid MPI/OpenMP Implementation

Remarks:
1. The parallel algorithm achieves linear scalability with a few number of processors;
2. The classification phase is sequential and takes a constant amount of time, thus affects the overall speedup of the algorithm;
3. Further work could be done to parallelize the boosting procedure in each classification phase.