Optimizing Performance of Financial Risk Calculations

Amit Kalele
Easwar Subramanian

March 19th, 2015
GPUs in Risk Problems

- Real Time Risk Management
  - Principal Component Analysis

- Incremental Risk Charge
  - Cumulative distribution

- Value at Risk
  - Portfolio (Re) valuation
PCA in RTRM

• Principal component analysis (PCA) – An important and compute intensive part of the RTRM application.

• PCA – Computes most important risk factors associated with a portfolio

• Bottleneck – Computation

• Solution – Parallel computations on Multicore systems
  • NVIDIA GPUs
  • Intel MIC
PCA Using DSYEVR

- Principal component analysis (PCA) – Eigenvalue and Eigenvector computations
- CLAPACK Algorithms
  - DSYEVD
  - DSYEVR
- Tri Diagonalization most compute Intensive
  - Consumes around 98% of time for partial spectrum
- Accelerate Tri Diagonalization
DSYTRD Routine

- CLAPACK Tri Diagonalization was completely re-coded using cuBLAS routines
- Minimum Data transfer between Host/GPU
Offloading Tri Diagonalization to GPUs

Input request

CPU/Host

Copy Matrix to GPU

Compute CLAPACK checks and constants

Compute eigenvalues/eigenvectors

GPU

Tri diagonalization

Compute House Holder Transformation

DsyMV

DsyR2

Copy Tri diagonal matrix to Host/CPU

Results
Results

DSYEV for 4500 assets

<table>
<thead>
<tr>
<th></th>
<th>Time in sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original CPU</td>
<td>85</td>
</tr>
<tr>
<td>MIC</td>
<td>11.5</td>
</tr>
<tr>
<td>K20</td>
<td>5.7</td>
</tr>
<tr>
<td>K40</td>
<td>4.4</td>
</tr>
</tbody>
</table>

Time in ms

Number of Assets

- GPU (ms)
- CPU (ms)
• Batch of 4000 PCA request – Sequential time is 677 sec
• Around 3000 request were of size < 1000
Comments and Further Scope

- Good scale up and performance for large size problems
- For assets < 700, **CPU performs better**
- Work in Progress
  - Block Reduction
    - Householder Transformations on Block of vectors
  - Possible to achieve $1.4x - 1.8x$ further gain
Incremental Risk Charge
IRC

To cover credit migration and default risk
- Additional capital charge
  - Incremental Risk Charge (IRC)
IRC Computations

- Estimate credit loss
  - Loss event – defaults, market crash, terror, calamity etc.
  - Severity of loss events – Impact
  - Cumulative effect of all the events

\[
Aggregated\ Loss\ distribution\ Z = \sum X_i
\]

\[X_i - \text{random variable}
\]

Severity of loss due to a random event & common distribution (function) \(F\), and \(f_i\) as probability density functions

- To estimate \(Z\)
  - Need to predict future random events with their severity of loss
  - Need to find the cumulative distribution \(F\)
IRC Computation

Estimate severities $X_i$

Calculate distribution $F$

Calculate cumulative distribution function for random variable $Z$

Simulate $X_i$ using Monte Carlo

Compute distribution function $F$ by possibly curve fitting on simulation results

Panjer recursions, FFTs etc.

FFTs are computational Bottleneck

$\{f_0, f_1, \ldots, f_{m-1}\}$ DFT $\rightarrow \{f^0, f^1, \ldots, f^{m-1}\}$ $\rightarrow$ INV-DFT $\rightarrow f$

FFTs are efficient for $M = 2$ powers, Cooley-Tukey Radix 2 Algorithm, $O(N\log N$ complexity)
FFTs in IRC

- FFTs for aggregation or compounding of default loss distributions
- Offloaded to a grid of 50 workstations
- Compute time – 45 min
- Consumes over 90% of total time
- Solution – Optimize FFT computations
IRC Calculations – Compute and Data intensive

- **Computations**
  - FFT computations for 133 scenarios
  - Each scenario consisting of 160,000 arrays
  - Each array with 32768 elements (*doubles*)

- **Data**
  - Each scenario translates into ~37GB of data
  - Total data – ~4.9TB
Performance Optimization on GPU

- Pinned Memory, Multiple streams, Hybrid (CPU+GPU)
Overlapping With Streams
Performance Optimization Intel MIC

- Thread binding, Memory Alignment, Reusing FFT descriptor, Symmetric or Hybrid computations

![Graph showing execution time in minutes for different optimizations.](image-url)
Performance on Various Platforms

**IRC Problem**

- **Execution time in min**
  - Original: 50 work Stations
  - MIC: 8.645
  - K40: 3.38

![Bar chart showing execution times for different platforms](chart.png)
Value at Risk
HVaR Computations

- Retrieve historical data and generate future scenarios based on historical data-
- Re-value each contract in the portfolio for every possible future scenario
- Contract values for a particular scenario are summed to obtain a portfolio value for that scenario
- Evaluate profit and loss (P&L) numbers by comparing current portfolio value and portfolio values obtained for each scenario
- Compute portfolio VaR from P&L values at the desired confidence level
• Historical VaR
• 4 million sensitivities (IR & FX gamma and delta)
• 1 year of historical data for each
• Serial execution time 63 sec
  • Significant when number of computations 500~1000
Performance Optimization

- Optimized serial code – $63 \rightarrow 8$ sec
- Shared memory, Data access coalescing, Multi Stream for overlap
- Hybrid Computing – aggregation in two stages
  - Massive reduction on GPU
  - 2nd stage on CPU while GPU reduces bigger array
Results

HVaR Computation for 4 million Sensitivities

- **Serial Optimal**: 8
- **Multicore CPU**: 0.344
- **K40**: 0.089

*Compute time*
Conclusions

• Real Time Risk Management
  • Principal Component Analysis
    • Tri Diagonalization offloaded to GPU – 19x

• Incremental Risk Charge
  • Cumulative distribution
    • FFT computations offloaded to GPU – 13x
    • Hardware reduction 50 workstations → 1 server with 2 GPUs

• Value at Risk
  • Portfolio (Re) valuation
    • (Re) pricing of portfolio – 89x
Sincere thanks to Pradeep Gupta
NVIDIA Pune, India
Thank You