ACCELERATING REINSURANCE ANALYTICS ON CLUSTERS, CLOUDS, AND GPU

Andrew Rau-Chaplin
arc@cs.dal.ca

Risk Analytics Lab,
Faculty of Computer Science
Dalhousie University,
Halifax, Canada

www.Risk-Analytics-Lab.ca
CAT RISK MARKET PARTICIPANTS
THE “CAT RISK ANALYTICS PIPELINE”

1. Cat Risk Assessment
   • Quantification of risk
   • Modeling of individual risks
   • Multiple views: Events and Accumulation

2. Cat Risk Management
   • Portfolio Management & Pricing of Risk
   • Understanding correlation
   • Optimization and decision making

3. Enterprise Risk Modeling
   • Dynamic Financial Analysis
   • Integration of risk from many sources
   • Optimization and decision making
Risk Analytics Lab

“Connecting Science and (Re)Insurance”

Risk Analytics

Algorithms, Statistics, Modeling, & Science

HPC: Clusters, Clouds, & GPU

Model: Collaborative Research with Industrial Partners

Key Outputs
- Highly qualified graduate students
- Industrial strength prototypes
- Peer reviewed publications
Risk Analytics Lab: People

- Postdocs, Ph.D. & MCS students, Visiting researchers & interns
- Disciplines
  - Computer Science
  - Statistics
  - Civil Engineering
  - Oceanography
  - Economics
  - Environmental Studies
  - GIS
  - Management Information Systems
High Performance Computing Can be Harnessed for Better, Faster Risk Analysis and Portfolio Management

**Reference Pipeline**

- Cat Risk Modeling
- Portfolio Management & Pricing
- DFA

**HPC Accelerated Pipeline** *(C/C++, MPI, OpenMP, CUDA)*

- Cat Risk Modeling
- Portfolio Management & Pricing
- DFA
RISK ANALYTICS LAB: PROJECTS...
EFFICIENT FRONTIER APPROACHES TO TREATY OPTIMIZATION

A Treaty Optimization Problem

- Optimization From a Primary Insurers or Broker’s Perspective!
- Find a Pareto Frontier

<table>
<thead>
<tr>
<th>Solution</th>
<th>Layer</th>
<th>Limit</th>
<th>Deductible</th>
<th>Placement</th>
<th>Max Recovery</th>
<th>Premium</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Layer 1</td>
<td>200</td>
<td>300</td>
<td>50%</td>
<td>120</td>
<td>$40</td>
</tr>
<tr>
<td></td>
<td>Layer 2</td>
<td>200</td>
<td>100</td>
<td>20%</td>
<td>40</td>
<td>$40</td>
</tr>
<tr>
<td>2</td>
<td>Layer 1</td>
<td>200</td>
<td>300</td>
<td>30%</td>
<td>60</td>
<td>$25</td>
</tr>
<tr>
<td></td>
<td>Layer 2</td>
<td>200</td>
<td>100</td>
<td>95%</td>
<td>190</td>
<td>$100</td>
</tr>
</tbody>
</table>
Inputs/Outputs

Discretization = 10%, 5%, or 1%
Existing Enumeration Approach

- Discretized search parameters
- Calculate results for all possible combinations of shares.
- Results in a large number of computations
  - Number of computations exponential increases with dimensions
    - Number of Layers
    - Number of share intervals
- Even parallelism is of limited help!
ROUND 1:

- Need a better algorithm!
  - Use an evolitional search approach
  - Population Based Incremental Learning → Di-PBIL

- Single risk measure (ie. 2D Pareto Frontier)
  - Variance
  - Value At Risk (VaR)
  - Tail Value at Risk (TVaR)

- Prototype in R (with mutlithreading)

- Questions
  - Quality: How close to the exact method?
  - Performance: How fast? How big a problem can we now handle?
**Quality: How close to the exact method?**

- Percentage of time DiPBIL finds the same solution as the exact method?

<table>
<thead>
<tr>
<th></th>
<th>Pop 100</th>
<th>Pop 200</th>
<th>Pop 400</th>
</tr>
</thead>
<tbody>
<tr>
<td>500 it</td>
<td>50%</td>
<td>74%</td>
<td>78%</td>
</tr>
<tr>
<td>1000 it</td>
<td>60%</td>
<td>76%</td>
<td>82%</td>
</tr>
<tr>
<td>2000 it</td>
<td>64%</td>
<td>88%</td>
<td>88%</td>
</tr>
</tbody>
</table>
**Quality: How close to the exact method?**

- Average error when DiPBIL does not find the same solution as the exact method?

Error always less than 6/100ths of a percent.
**Performance: How fast when compared to the exact method?**

- Time on a single core to compute a single point on efficient frontier for 7 layers and 5% discretization

![Graph showing time comparison between Enumeration, Pop 100, Pop 200, and Pop 400 methods.](image)

- Enumeration: weeks
- Di-PBIL: 2-15 minutes
**Performance: How Big a Problem Can We Now Solve?**

- Time on a single core to compute a single point on efficient frontier at 5% discretization.

![Graph showing solutions times no longer exponential in the number of layers](chart.png)
ROUND 2:

- Single risk metric → **Multiple** risk metric (e.g. 1 in 100yr TVaR + 1 in 5yr VaR)
- 2-d Pareto front → 3-d+ Pareto front
- Di-PBIL → **Mo-PBIL**
- Prototype in R → **Prototype in C++**

**Advantages**
- Search for whole front, not point by point
- Multiple Risk Metrics
- Performance!
ROUND 2: OPTIMIZED Mo-PBIL

Mo-PBIL: Complete frontier (60 - 70 points) for 7 layer program and 5% discretization in 16 seconds!

Setup: 500 iterations, 128 population
2 * Xeon E5-2660 processors
SUMMARY

- Evolutionary techniques work well for Treaty Optimization!
- **Can now solve practical problem instances with practical performance.**
- Compared multiple evolutionary search methods
  - Single Objective: DE, PSO, GA, PBIL
  - Multi Objective: VEPSO, MODE, SPEA2, NSGA2
- Evaluation Results
  - All work and can produce high quality solutions
  - Differences
    - Easy of use
    - Performance
CURRENT WORK

Optimize layer structure, not just shares!

Inputs

- Exh
- Att
- Discretization = d%
- Risk Measure
- Premium function
- # reinstatements
- Aggregate terms (ie 3rd event cover)
- Set of ELTs
- 100K Year Event Table (YET)

Treaty Optimizer 2

Aggregate Simulation Engine

Risk

Expected Return
PARALLEL SIMULATIONS FOR ANALYSING PORTFOLIOS OF CATASTROPHIC EVENT RISK

Risk Analytics Lab, Dalhousie University, Halifax, Canada
www.Risk-Analytics-Lab.ca

Additional work by Neil Burke
**Reinsurance Analytics Pipeline**

- **Risk Quantification** → **Portfolio Management and Pricing**

- **Exposure**
  - 10K * 1M

- **Cat Models**
  - 100s

- **Modeled ELTs**

- **Portfolio Defs**

- **Treaty Terms**

- **Year Event Table (Simulated Future)**

- **Pricing**

- **Portfolio Analysis**

- **Dynamic Financial Analysis**

- **Optimization**

- **Visualization**

- **Coverage** → **Time**
ANALYSIS: PER OCCURRENCE AND AGGREGATE FEATURES

INPUTS
- Year Event Table
- Event Loss Table
- Layers

Aggregate Risk Engine (ARE)

OUTPUTS
- Loss per Trial Per Layer
Inputs: Event Loss Table (ELT)

A Program ≈ Multiple layers, over ~15 ELTs, covering ~5 models, and ~200K events

A Portfolio ≈ 3-4K Programs each with multiple layers, with 40K ELTs, over 100 models, covering 1M events

Event Loss Table (ELT)

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Annual Rate</th>
<th>Mean</th>
<th>St. Dev. Ind.</th>
<th>St. Dev. Corr</th>
<th>Exposure</th>
</tr>
</thead>
<tbody>
<tr>
<td>469292</td>
<td>0.0029%</td>
<td>226,945</td>
<td>114,932</td>
<td>28,733</td>
<td>11,347,250</td>
</tr>
<tr>
<td>469282</td>
<td>0.0002%</td>
<td>213,863</td>
<td>111,154</td>
<td>27,789</td>
<td>10,593,150</td>
</tr>
<tr>
<td>469293</td>
<td>0.0123%</td>
<td>298,009</td>
<td>109,865</td>
<td>16,765</td>
<td>10,350,459</td>
</tr>
<tr>
<td>469266</td>
<td>0.0004%</td>
<td>345,678</td>
<td>126,726</td>
<td>32,766</td>
<td>21,814,950</td>
</tr>
<tr>
<td>469283</td>
<td>0.0439%</td>
<td>196,299</td>
<td>100,123</td>
<td>26,681</td>
<td>12,114,321</td>
</tr>
</tbody>
</table>
INPUTS: FINANCIAL TERMS

Exposure

Cat Models

ELTs

A Program

Multiple Layers

Layer 1
Layer 2
Layer 3

Layer 3

200M Limit

75M Limit

25M Retention

25M Retention

Occurrence Terms

Aggregate Terms

Plus...

- Reinstatements,
- Inuring,
- Other financial terms
**INPUTS: Event Catalog & Year Event Table**

### Event Catalog

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Annual Rate</th>
<th>( \sigma )</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0439%</td>
<td>EQ</td>
<td></td>
</tr>
<tr>
<td>469,292</td>
<td>0.0029%</td>
<td>FL</td>
<td></td>
</tr>
<tr>
<td>1,000,000</td>
<td>0.1402%</td>
<td>T</td>
<td></td>
</tr>
</tbody>
</table>

### Year Event Table (YET)

<table>
<thead>
<tr>
<th>Year (Trial)</th>
<th>Seq. No.</th>
<th>Event ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>437</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>769,251</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>469,292</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1M</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ROUND 1: ALGORITHM SKETCH

For each Program P
  For each Layers L in P
    For each Trial T in the YLT
      For each Event in T
        For each ELT covered by L
          Lookup event E in the ELT and find the corresponding loss X
          Apply Financial Terms to X
          Prefix sum over all occurrences in T
          For each occurrence in T
            Apply Financial Terms to x
            Reverse cumulative sum over all occurrences in T
            Add it to the occurrences sum
          For each occurrences in occurrences sum
            Apply layer attachment
            Prefix sum over all occurrences in occurrences sum
            For each occurrences in occurrences sum
              Apply Layer Limit
              Reverse cumulative sum over all occurrences in occurrences sum
              Add up all the losses to report the final result
HARDWARE

- Intel Xeon 5E-2660 (8 core)

- Nvidia C2075 GPU Computing Processor

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Model</th>
<th># of GPUs</th>
<th>Core clock in MHz (each)</th>
<th>Shaders</th>
<th>Memory</th>
<th>Processing Power (peak) GFLOPs[^2]</th>
<th>Compute capability</th>
<th>TDP watts</th>
<th>Form factor and features</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2050/C2070/C2075 GPU Computing Processor</td>
<td>C2050/C2070/C2075</td>
<td>1</td>
<td>575</td>
<td>440</td>
<td>144</td>
<td></td>
<td>1286</td>
<td></td>
<td>2.0</td>
</tr>
</tbody>
</table>
PERFORMANCE OF THE BASIC AGGREGATE ANALYSIS ALGORITHM ON A CPU USING A SINGLE CORE (i7)
AGGREGATE ANALYSIS ALGORITHM ON A MULTI-CORE CPU (i7)

- OpenMP Parallel For over Trials loop
- Careful management of public/private variables
- Optimization for vector units

- Limited speedup!
- 2 cores $\rightarrow 1.5x$
- 4 cores $\rightarrow 2.2x$
- 8 (virtual) cores $\rightarrow 2.6x$

- Many threads helps hide memory latency
- 256 threads per core $\rightarrow 7.4\%$ improvement
- More threads per core does not really help.
CHALLENGES

- Not much floating point!
- The issues is random access to ELTs
- Need more bandwidth to memory
- What about GPUs?
Basic GPU Algorithm

- One thread per trial
- Event Loss sets in Global device memory
- Event sets stored as direct access tables
- Shared memory as working space
- How many threads should be assigned per CUDA block?
AGGREGATE ANALYSIS ALGORITHM ON MANY-CORE GPU: THREADS PER CUDA BLOCK

- Trade off
- Example: 1M threads for 1M trials & 256 threads per processor ➔
  - 3906 blocks over 14 processors, for
  - 279 blocks per processor
- More threads per block
  - Better hiding of latency to global memory
  - Less space per thread in shared memory
**Optimized Aggregate Analysis on a Many-Core GPU**

<table>
<thead>
<tr>
<th>Trial</th>
<th>Event</th>
<th>ELT1</th>
<th>ELT2</th>
</tr>
</thead>
<tbody>
<tr>
<td>77</td>
<td>99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>90</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>93</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>13</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Ideas**
- Chunk up the computation
- Fetch losses associated with a chunk into shared memory and work with them there
- Balance chunk size and thread count
- Reorganize ELTs to promote striding
- ...
Optimized Aggregate Analysis on a Many-core GPU: Chunking

![Graph showing time (sec) vs. size of chunk for Variant Implementation on Many-core GPU. The graph peaks at a size of 16 and shows a decrease and then increase as the size of the chunk changes.](image-url)
Goals

- Add support for secondary uncertainty
- Speed up code to support portfolio analysis
  - Vectorize the computation for multicore processors
  - Add support for Intel Phi
  - Hybrid solutions
ROUND TWO: ON MULTICORES

- Good News: Optimization/Vectorization helps on Multicore Proc

<table>
<thead>
<tr>
<th>Per Layer</th>
<th>100K Trials</th>
<th>800K Trials</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous Version</td>
<td>13.2 seconds</td>
<td>123 seconds</td>
</tr>
<tr>
<td>New Version</td>
<td>1.6 seconds</td>
<td>12.9 seconds</td>
</tr>
</tbody>
</table>

- Adding Secondary uncertainty
  - Development of fast beta inverse code

<table>
<thead>
<tr>
<th>Per Layer</th>
<th>100K Trials</th>
<th>800K Trials</th>
</tr>
</thead>
<tbody>
<tr>
<td>With Secondary Uncertainty</td>
<td>5.9 Seconds</td>
<td>47.02 Seconds</td>
</tr>
</tbody>
</table>
ROUND TWO: MULTICORE + NVIDIA C2075 GPU

- Adding Secondary uncertainty
  - Development of fast beta inverse code

<table>
<thead>
<tr>
<th>Per Layer</th>
<th>100K Trials</th>
<th>800K Trials</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Secondary Uncertainty</td>
<td>2.1 seconds</td>
<td>17.57 seconds</td>
</tr>
<tr>
<td>With Secondary Uncertainty</td>
<td>3.5 Seconds</td>
<td>55.1 Seconds</td>
</tr>
</tbody>
</table>
SUMMARY

- The bottleneck is bandwidth to memory!! (not computation)
- Performance number for a 2 socket server (<$8K)

Practical for Pricing Use Case
(Typical Program = 3 layers, 15 ELTs, 1000 events/trial, no secondary uncertainty)
- 800K trial aggregate simulation in 38.4 seconds, or
- 100K trial aggregate simulation in 4.8 second → real-time pricing!

Portfolio Analysis Use Case
(Typical Portfolio = 1000 programs with 3 layers each, 15 ELTs, 1000 events/trial)
- 100K Aggregate analysis on in 1.3 hours
- 800K Aggregate analysis on in 10.6 hours
- Multiple GPU/CPUs needed for practical 800K trial aggregate simulation on typical portfolio but the methods scale easily!

- Multicore vs GPU? – Still an open question for this application!
QU-PARA:
QUERY-DRIVEN LARGE-SCALE PORTFOLIO AGGREGATE RISK ANALYSIS ON MAPREDUCE

www.Risk-Analytics-Lab.ca

A. Rau-Chaplin, B. Varghese, D. Wilson, Z. Yao, N. Zeh,
"QuPARA: Query-Driven Large-Scale Portfolio Aggregate Risk Analysis on MapReduce",
Production Analytics Systems

- Very Fast
- Standardized
- Small # of standard analyses
- Small # of key outputs
- Coded by programmers, slow to change
- Closed, Auditable

Exploratory Analytics Systems

- Fast
- Flexible
- Uncountable # of Ad Hoc analyses
- Huge # of possible outputs
- Coded by analysts, quick to change
- Open
Yes, but actuaries and underwriters have lots of other questions!

**Wanted:** A framework for exploring risk portfolios

**Approach:** Applying Big Data techniques to Reinsurance analytics

**Goal:** A flexible MapReduce framework for portfolio risk analysis that facilitates solving a rich variety of catastrophic risk queries in a timely manner.
Examples of “Natural but Ad Hoc” Queries
STANDARD AGGREGATE ANALYSIS

- Basic Portfolio Analysis
- Loss Distribution
- Exceedance Probability

<table>
<thead>
<tr>
<th>Sig. Level</th>
<th>Return Period</th>
<th>VaR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.004</td>
<td>1/250</td>
<td>65</td>
</tr>
<tr>
<td>0.01</td>
<td>1/100</td>
<td>57</td>
</tr>
<tr>
<td>0.05</td>
<td>1/20</td>
<td>42</td>
</tr>
<tr>
<td>0.1</td>
<td>1/10</td>
<td>35</td>
</tr>
<tr>
<td>0.2</td>
<td>1/5</td>
<td>30</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Losses by Treaty Terms or Region/Peril Coverage

- Multi-Loss Distribution

Diagram:
- Portfolio
- YET
- ELT
- Analysis Process
- Region, Peril Selection or LOB, COB, TOP Filter
- Exposure Region or Peril Selection

Graph:
- Loss Value Distribution
- Probability Density
- Loss Value (Million $)
- US HU
- CA HU
- US EQ
PERIODIC LOSS DIFFERENCES

- Time based loss analysis

![Diagram showing seasonal loss value distribution and expected losses for Q1, Q2, Q3, and Q4.]

- Portfolio
- YET
- ELT

- Time Interval
- Analysis Process
MULTI-MARGINAL ANALYSIS

Contract Layer Combination

Portfolio (PF) + VaR (0.05)

<table>
<thead>
<tr>
<th>Portfolio (PF) +</th>
<th>VaR (0.05)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF</td>
<td>10</td>
</tr>
<tr>
<td>PF+T1</td>
<td>11</td>
</tr>
<tr>
<td>PF+T2</td>
<td>13</td>
</tr>
<tr>
<td>PF+ T3</td>
<td>12</td>
</tr>
<tr>
<td>PF+T1+T2</td>
<td>11.5</td>
</tr>
<tr>
<td>PF+T1+T3</td>
<td>12</td>
</tr>
<tr>
<td>PF+T2+T3</td>
<td>16</td>
</tr>
<tr>
<td>PF+T1+T2+T3</td>
<td>18</td>
</tr>
</tbody>
</table>

Diagram:
- T1, T2, T3
- Portfolio
- YET
- ELT
- Analysis Process
- f

Table:
- Portfolio (PF) + VaR (0.05)
- PF: 10
- PF+T1: 11
- PF+T2: 13
- PF+T3: 12
- PF+T1+T2: 11.5
- PF+T1+T3: 12
- PF+T2+T3: 16
- PF+T1+T2+T3: 18
Basic Map-Reduce Model for Aggregate Analysis

- M/R Model for Aggregate Analysis
  - Map: Loss calculation
  - Combine: Loss grouping
  - Reduce: Loss aggregation

Diagram:
- HDFS
  - Distributed Cache
  - Portfolio Data & ELT
  - YET
  - Result
- Mapper
- Combine
- Reduce
- Result
HADOOP + HIVE REALITY

Diagram illustrating the integration of Hadoop and Hive in a real-time data processing environment.
“HIGH PERFORMANCE” JAVA

- **General Tricks**
  - Using binary representation for intermediate results
  - Using java NIO API to do fast I/O operations

- **The need for memory efficient data structures**
  - Standard Java Collection hash tables 10x size overhead
  - Trove Hash tables 5x size overhead
  - Sorted Arrays + binary search 1.1x size overhead + small access time overhead

- **Java Runtime**
  - Optimize JVM heap size
  - Utilize JIT optimization
  - Optimize GC parameters
Experimental Performance – Time/Speedup

- Time/Speedup Testing
  - 1M trial simulation on 3200 layers with between 16 and 256 cores
  - On 256 cores
    - ~21.7 minutes, 84% speedup

![Graph showing time test with fixed 3200 layers and increasing number of cores (16 to 256).](image)
**Experimental Performance — Size Up**

- **Experiment:** Increasing the amount of layers (from 200 layers to 3200 layers) keeping cores = 256

- **Result:** Significant overhead time on small data, time per layer decreasing
**Experimental Performance - Scale Up**

- **Experiment:** Increasing # of cores (from 16 to 256) but keep the work amount per core (200 layers per 16 cores) as the same.

- **Result:** The time difference between 16 cores and 256 cores is less than 8% of the total time.
INTEGRATION AND REPORTING

Pentaho
- Dashboards
- Job Flow
- Reporting
**SUMMARY**

- Large scale Risk Analytics based on stochastic simulations can be practical in Hadoop
- Not trivial to get performance in Java/Hadoop
- Ad hoc Hadoop analysis job != A Framework

**FUTURE WORK**

- Location level analysis - 100,000 times the data?
- Exposure Management – need to add spatial analysis
- Suggestions?
**Risk Analytics Lab:**
**Seeking New Industrial Collaborators!**

"Connecting Science and (Re)Insurance"

Risk Analytics

HPC: Clusters, Clouds, & GPU

Algorithms, Statistics, Modeling, & Science

Andrew Rau-Chaplin
arc@cs.dal.ca

Risk Analytics Lab,
Faculty of Computer Science
Dalhousie University,
Halifax, Canada

www.Risk-Analytics-Lab.ca