The fast and the fabulous

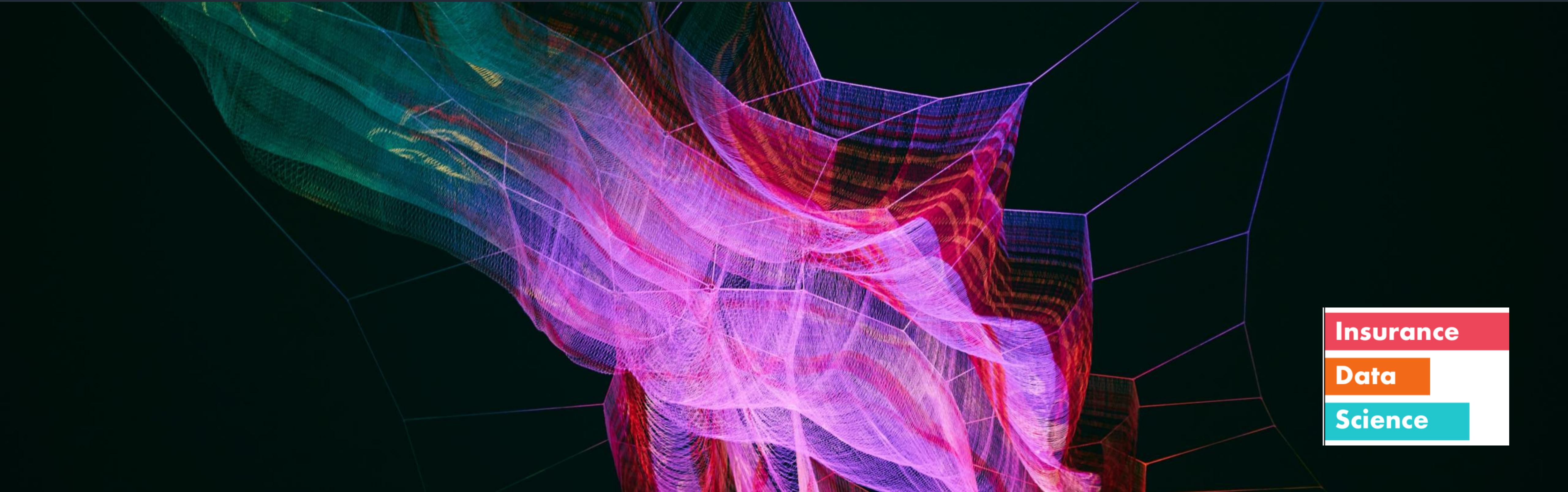
Harnessing GPU power for high-performance life insurance computations.

Karol Maciejewski

JUNE 16, 2023

INSURANCE DATA SCIENCE CONFERENCE

15-16 JUNE 2023, LONDON



Agenda (Table of contents)

This presentation is partially based on a Milliman research paper:
Maciejewski K., Echchelh M., Sznajder D., ‘Building a high-performance in-house life projection and ALM model: architecture and implementation considerations in Python’, 2023
https://www.milliman.com/en/insight/building-in-house-projection-alm-model-python

Life insurance projection models

Liabilty Cash Flows and Asset-Liability Models

Cashflow projection models – cornerstone of modern life insurance modeling:

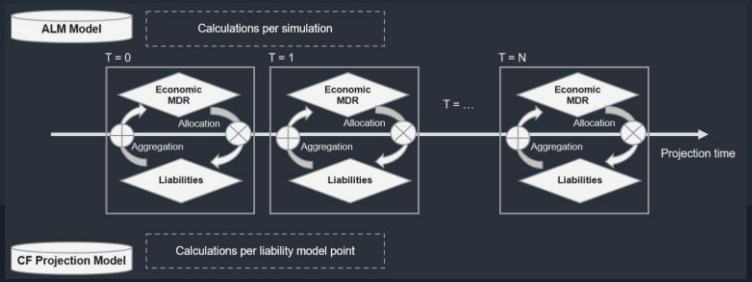
BEL & MV balance sheet projections	Solvency II & capital projections	IFRS 17	Pricing & profit test	Business planning	Valuation and M&A

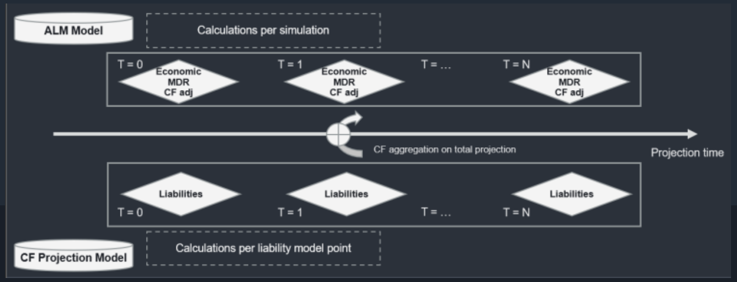
Typically, a modern ALM model has the following key building blocks and characteristics:

Liability CF projection	Asset side (ALM)	A ←→ L link
Policy/Model point level	Asset model point level	Dynamic (full ALM)
Deterministic (expected value)	Stochastic (w.r.t. economic env)	Flexing
Decrements, cashflows	Asset and portfolio strategy	
Statutory reserves (technical/MGR)	Discretionary management rules	

Complexity is (usually) not in mathematical sophistication, but in the high dimensionality of these models.

Full ALM vs Flexing:





GPU architecture and computing fundamentals

CPU vs GPU and GPU implementations in Python

Central Processing Unit (CPU)

- Versatile calculation engine at the heart of any electronic device nowadays
- Very good at heterogenic computations (MIMD**) and switching tasks
- Very fast single-core computations
- Usually with a limited number of cores that allow parallel computations for software supporting it

** For proper definitions and good paper on parallel computing see:
Flynn M. "Very high-speed computing systems", Proc. of IEEE, 1966

Graphical Processing Unit (GPU)

- Designed for accelerated graphical computations for computer games, but evolved to handle more generalized workloads (CUDA, OpenGL)
- GPU computation power at the basis of cryptocurrency and recent AI headway
- Allows highly parallel homogenic computations (SIMD**)
- Single mid-tier household level GPU can have 5000+ cores and high-tier cards can have double
- Individual cores less powerful than CPU, but power is in the numbers
- Parallel computing comes with a cost, some additional considerations: synchronization, race conditions, …

Model Implementations on GPU

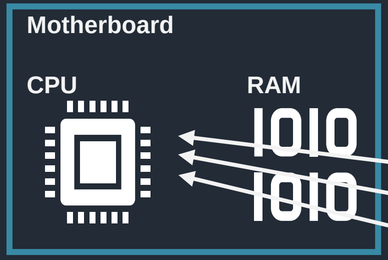
- The first language for GPU computations was CUDA C++ from NVIDIA
- Similarly to C++, very performant, but low level with high entry costs (and not widespread in actuarial world)
- There are numerous libraries in higher level languages, such as Python, that allow using GPU without all the technical bacground
- Some utilize GPU for specific purposes completely transparent to the users (e.g. ML libraries like Keras, Tensorflow, …)
- Some give access to a set of generic functions that can be used in any way by the users (e.g. CuDF, CuPy libraries from Rapids AI)
- Some give the user broad access to GPU capabilities (e.g. Numba, PyCUDA), but with great power comes the need of deeper understanding of GPU architecture and working

GPU architecture and computing fundamentals

GPU architecture and data processing

Central Processing Unit (CPU)

- Iterate over the data cells and apply calculation (sequential calculation)



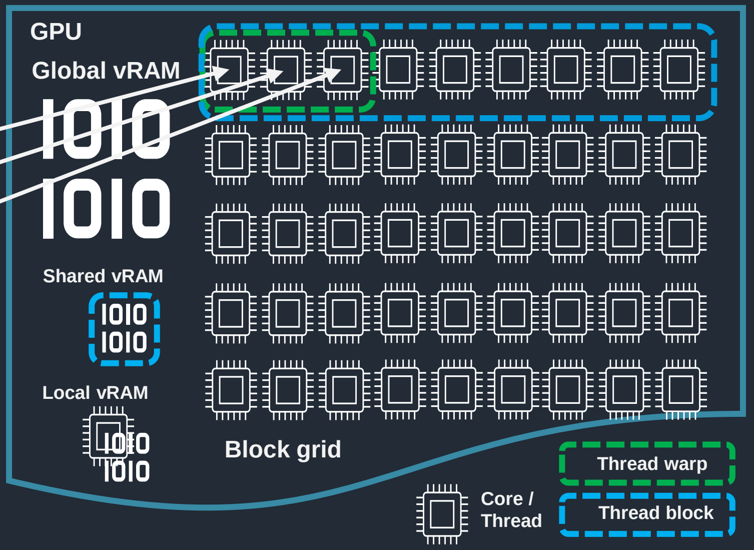
Data for computations

- For best GPU benefits the data should be big enough and the calculations should be similar, independent and without too many logical branches

- Certain types of calculations parallelize better than others (e.g. map functions vs reduction functions)
- Understanding how GPU works is key to efficient GPU implementations and custom model design

Graphical Processing Unit (GPU)

- Choose dimension(s) for parallelization and assign each data cell to a GPU thread along that axis (parallel calculation)



- GPU threads organized in blocks and grid

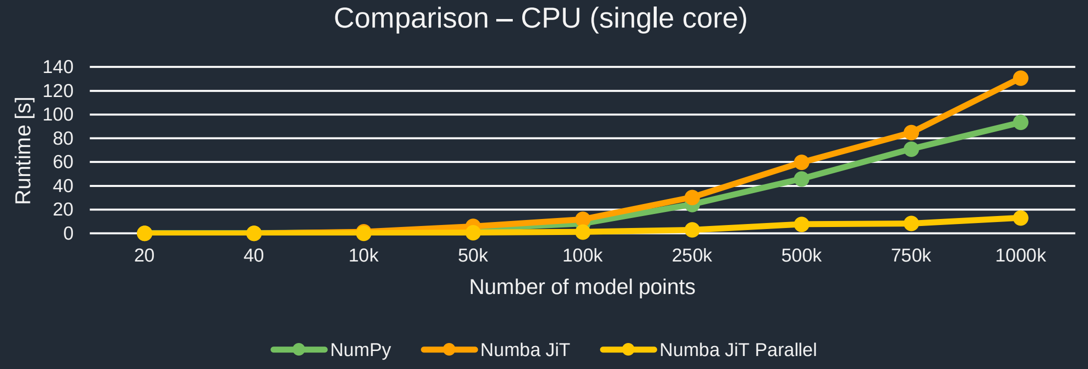
Selected benchmarks

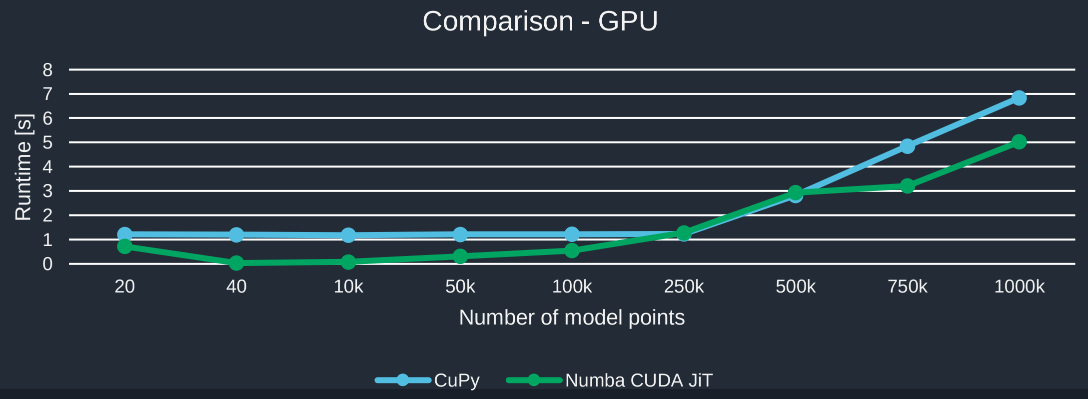
Projection model performance comparison

Liability cashflow projection model

Model from the research paper with simple saving product.

- 20x performance improvements for large samples

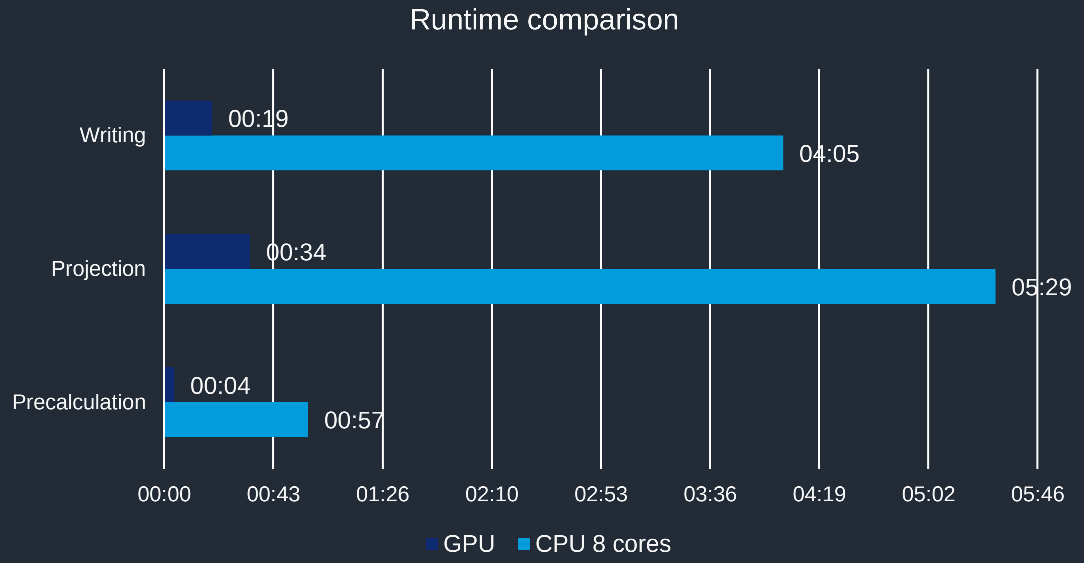




Flexing ALM model

Model from a client proof-of-concept implementation with 3 types of liability products and complete asset model and asset-liability strategy, based on flexing. Typical EoY data sizes and projection parameters (stochastic ALM).

- 10-15x performance improvements to 8 core, 80-120x to single core



Milliman

Thank you

Karol Maciejewski

karol.maciejewski@milliman.com