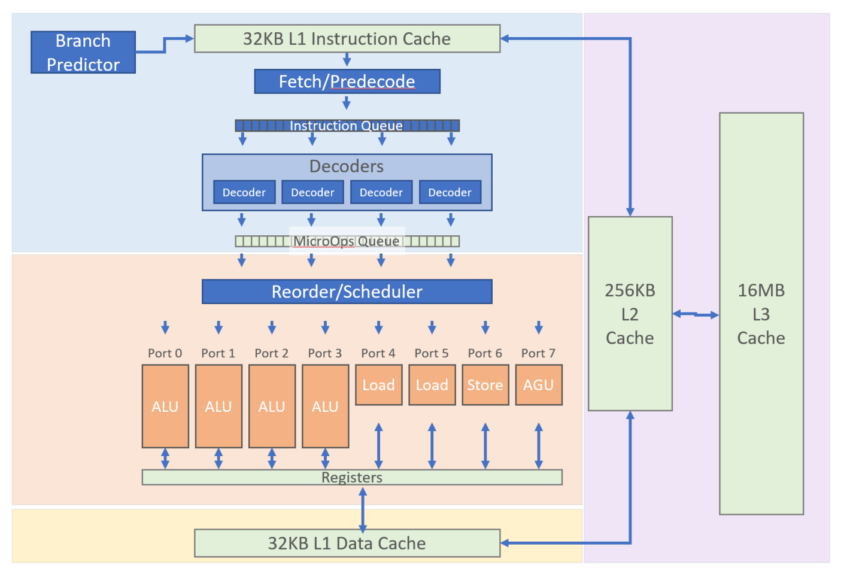
nZetta Derivatives Pricing Toolkit: Monte Carlo

Jason Charlesworth (Zettamatics & NAG)

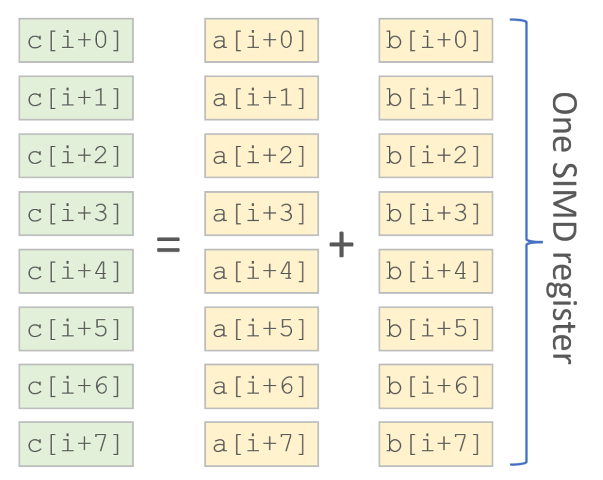
Monte Carlo: What's the Problem?

Monte Carlo (MC) methods are widely used for pricing complex, exotic derivatives. However, MC simulations can be computationally expensive and noisy resulting in inaccurate risk calculations.



To achieve optimal performance and accuracy, it is crucial to leverage the full potential of hardware parallelism: exploiting the features of x86 architecture such as superscalar, SIMD, and threading, as well as harnessing the power of GPUs for massive parallelism.

However, **parallel programming is challenging and requires careful algorithm design** and implementation.



To ensure fast and smooth MC risk calculations, Adjoint Algorithmic Differentiation (AAD) is *the* key technique. AAD allows for efficient computation of sensitivities and gradients, which are essential for risk management and hedging strategies.

nZetta Toolkit

The nZetta Toolkit is a C++17 library (Python API also available) containing all the performance-critical functionality needed for the fastest quant calculations. It abstracts the hardware so you can write your code once and it will run on x86, distribute across threads, or run on a GPU with just a single software switch. All functions support, and are optimized for, AAD.

Psuedo/Quasi Random Numbers:
- Mersenne, L'Ecuyer, Sobol
- Correlation and Repair (Cholesky, PCA, Rectangular)
- Brownian Bridge
- Antithetic
- MC-Engine (tranching)

Interpolation:
- Uniform/Non-Uniform/Multidimensional
- Range of extrapolations & interpolations

Tensor shaping: Concatenate, Broadcast, Repeat, Extract, Insert, Append, Transpose…

Reductions: Sum, Mean, Var, Min, Max, Prod

Regression: Linear, Quadratic, Loess

Windowed Reductions e.g., vol targets

Processes: (Quanto/shifted) Log-Normal, CIR, OU …

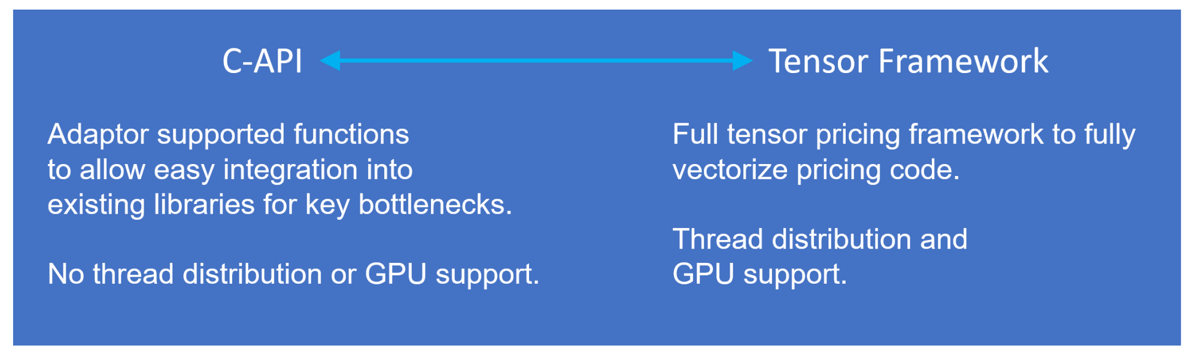
Finance Functions: SABR, BS Options, BS Implied vol, ND2, ND3 …

Multifactor PDE: Douglas ADI, Craig-Sneyd, Modified Craig-Sneyd, Hunsdorfer-Verwer

1,2,3-F Discontinuous Convolutional Lattices (2023Q4)

Using the nZetta Toolkit you can **achieve orders of magnitude improved pricing and risk**. You can position yourself to easily migrate to the latest hardware and AAD when needed.

nZetta Toolkit: Tensor Framework



Adaptor supported functions to allow easy integration into existing libraries for key bottlenecks.

No thread distribution or GPU support.

Full tensor pricing framework to fully vectorize pricing code.

Thread distribution and GPU support.

nZetta Toolkit functionality can be used via C-API or a tensor framework. Think C++ NumPy for finance.

Working on entire tensors of states allows efficient, large, blocks of vectorized work or task offloading to GPUs.

`std` maths operations, conditionals, as well as finance-specific operations are supported naturally, applying to entire tensors enabling full benefit from vectorization. Algorithms are designed for maximum performance in the finance domain.

```
// Evolve all s and update range-accrual
Tensor<double> phi = rng(npaths);
Tensor<double> vol = vols(s) * sqrtDt;
s                  = s * exp(r*dt + vol * (-0.5 * vol + phi));
IF((s >= slow) && (s <= shigh))
{
  ra = ra + 1.0;
}
```

```
// Run code in single-thread x86
setHardware(HardwareMode::x86);

// Distribute tasks across threads
setHardware(HardwareMode::x86,SynchMode::Asynchronous);

// Offload tasks to GPU
setHardware(HardwareMode::GPU);
```

Hardware use is abstracted; write the quant code once and a single software switch determines where the tasks are evaluated, whether without threading, distributed across worker threads, or using a GPU. With hardware abstraction, you can concentrate on finance and NAG provides the fastest performance.

The nZetta Toolkit: Quants focus on the finance, and nZetta Toolkit will provide the performance.

Example: Multidimensional FX Derivative Pricing

Options on multiple currencies are particularly hard to price using realistic FX dynamics. To capture the market-implied correlation structure, the model must accurately reproduce the observed prices of vanilla options on all relevant FX pairs. A standard model for this is **Heston Local Correlation** (HLC) model of De Col & Kuppinger.

The HLC model is a log-normal model with a stochastic covariance matrix driven by Heston processes. The lower-triangular matrix, $\mathbf{\Pi}$, is the Cholesky decomposition of the covariance matrix, Π^2 which uses state-dependent local-volatility correction surfaces, $A_{ij}(S)$, to calibrate to vanilla options.

$$\begin{aligned} d\mathbf{S}(t) &= \mathrm{diag}(\mathbf{S}(t))\big(\mathbf{a}(t)dt + \mathbf{\Pi}(\mathbf{S}(t),\mathbf{v}(t))d\mathbf{W}(t)\big) \\ dv_i(t) &= \kappa_i(\mu_i - v_i(t))dt + \alpha_i\sqrt{v_i(t)}dZ(t) \end{aligned}$$

$$\begin{aligned} \mathbf{\Pi}^2_{ii}(\mathbf{S}(t),\mathbf{v}(t)) &= A^2_{i0}(S_{i0},t)v_{i0}(t) \\ \mathbf{\Pi}^2_{ij}(\mathbf{S}(t),\mathbf{v}(t)) &= \frac{1}{2}\Big(A^2_{i0}(S_{i0},t)v_{i0}(t) + A^2_{j0}(S_{j0},t)v_{j0}(t) - A^2_{ij}(S_{i0}/S_{j0},t)v_{ij}(t)\Big) \end{aligned}$$

The Roadblock

The HLC model does **not** guarantee $\mathbf{\Pi}^2$ is positive-definite; for <u>every</u> timestep and for <u>every</u> Monte Carlo path, Π^2 must be examined, repaired (if necessary), and converted into its Cholesky decomposition and finally used to correlate the $N[0,\sqrt{dt}]$ Wiener processes, $d\mathbf{W}$.

Repairing a correlation matrix, ρ, using the Rebonato method involves calculating the eigenvalues and eigenvectors of the putative correlation matrix to create a valid matrix, $\tilde{\rho}$

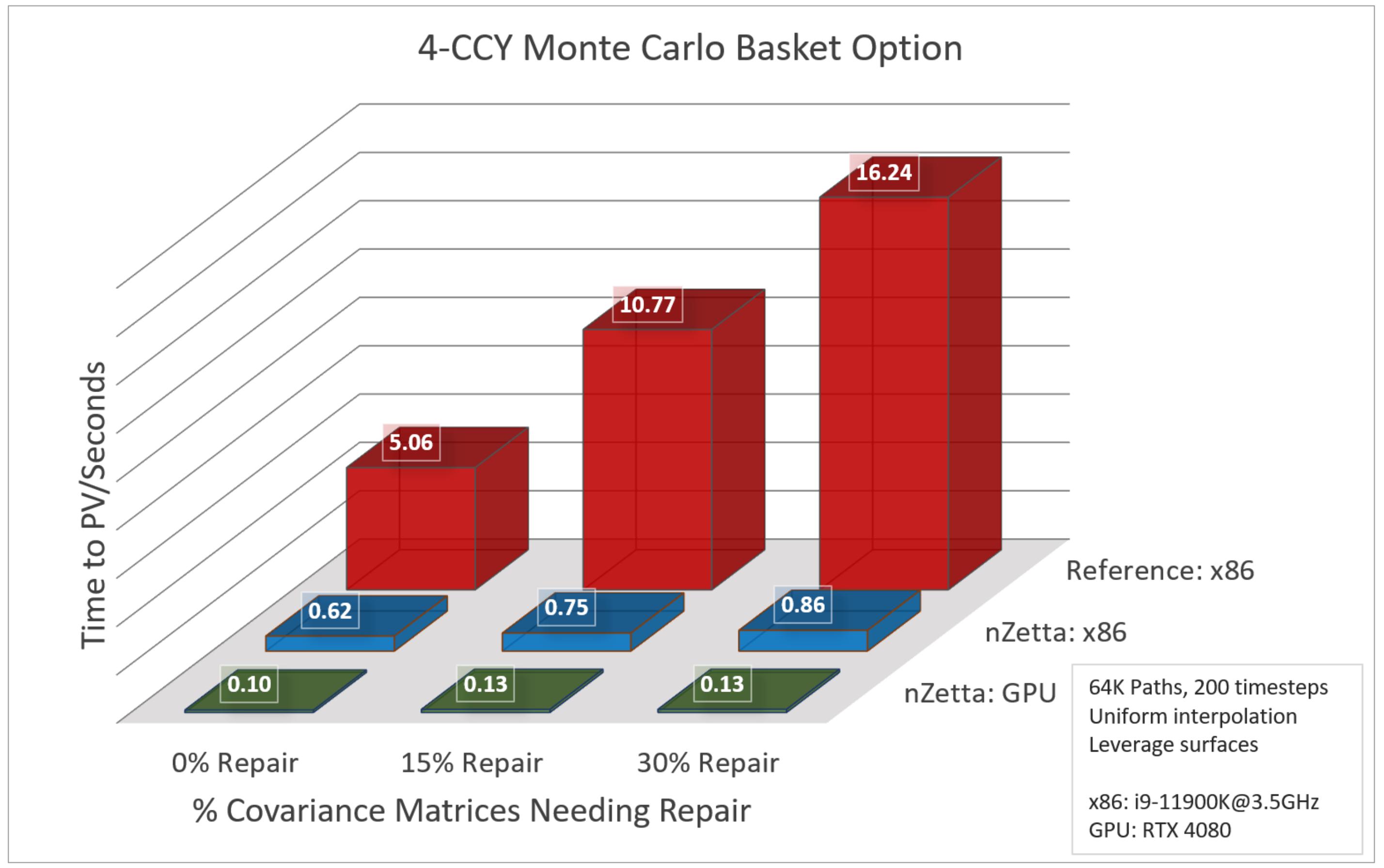
$$\rho \cdot \vec{e}^{(i)} = \lambda_i \vec{e}^{(i)} \quad \Rightarrow \quad \tilde{\rho}_{ij} = \sum_k \max(\lambda_k, 0) e^{(k)}_i e^{(k)}_j \alpha_i \alpha_j$$

Matrix libraries are optimized for *large* matrices. In finance, we require eigenvectors on *large* numbers of *small* matrices. Standard spectral decomposition approaches do not lend themselves to vectorization.

nZetta Toolkit & HLC

The nZetta Toolkit implements highly optimized methods designed for the finance domain. Critical to the Heston Local Correlation model, low-rank eigenvalue/vector calculations, Cholesky decomposition, correlation and interpolation.

The nZetta Toolkit achieved **over 15× speed-up** on single-threaded x86 and **over 120× speed-up** on low-end GPU using the **same** Quant code.



The nZetta Toolkit: Fastest performance whatever the platform. Write your model using the nZetta Toolkit, and run with x86 single-thread, x86 thread-distributed, GPU, and Algorithmic Differentiation.