## Introduction

**Current Situation**
- About 73% of US Equity Market Trading is via AT
- Expected to rise 10% over the coming 5 years
- HFT, dependent on low latency, high algorithmic efficiency, high speed trading
- HFT constitutes 50% of trading volume in US
- Constant competition to be microseconds faster
- De Prado mentions vectorization, multiprocessing, data manipulation among some ways to enhance HFT algorithms

**Advancement in Technology**
- Emergence of more ARM based processors in PCs and HPC (eg. Apple M Series, AWS Graviton2, Fugaku, Project Mont Blanc, etc.)
- Claims of higher performance and lower cost compared to x86
- Claims of better energy-to-solution compared to x86
- Inclusion of more optimization tools on ARM chips (eg. NEON)

**Capitalizing on the growth**
- HFT algorithms rely heavily on low latency solutions
- With more optimization tools available on ARM, HFT applications might have lower latency on ARM
- Specifically optimize HFT application to ARM architecture computer chips
- Present a more efficient backbone for HFT firms to carry out their operations

### NEON Intrinsics

NEON intrinsics are a set of instructions for Single Instruction Multiple Data (SIMD) operations.

<table>
<thead>
<tr>
<th>Intrinsic</th>
<th>Use case</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>vmlaq,f32</td>
<td>Multiply a float32 vector against a scalar value</td>
<td>( {a, b, c, d} \times {x, y, z, w} )</td>
</tr>
<tr>
<td>vsdq,f32</td>
<td>Divide a float32 vector by another float32 vector</td>
<td>( {a, b, c, d} \div {x, y, z, w} )</td>
</tr>
<tr>
<td>vsaddq,f32</td>
<td>Add all values up in a float32 vector</td>
<td>( {a, b, c, d} + {x, y, z, w} )</td>
</tr>
<tr>
<td>vsubsq,f32</td>
<td>Subtract a float32 vector form another float32 vector</td>
<td>( {a, b, c, d} - {x, y, z, w} )</td>
</tr>
<tr>
<td>vdupq_n_f32</td>
<td>Duplicate a float32 value across the 4 lanes</td>
<td>( {x, y, z, w} \times {x, y, z, w} )</td>
</tr>
<tr>
<td>vgenq_lane_n_f32</td>
<td>Get the value stored in the specified lane in the provide float32 vector</td>
<td>( {a, b, c, d} \times {x, y, z, w} )</td>
</tr>
</tbody>
</table>

**FYP22024 - The first implementation of an optimized HFT algorithm on an ARM chip**

### Implementation

**Application of NEON intrinsics on derivatives pricing:**
- Black Scholes Pricing for European Vanilla Option Pricing
- Monte Carlo Simulation using Gaussian Box Muller Algorithm
- Optimizing Existing Project on Derivative Pricing

**Key Takeaway**
- Vectorizing code using NEON intrinsics can reduce computation time by 15-82% depending on implementation and identification of bottlenecks.
- Low latency firms should invest more in ARM architecture, SWE with SIMD NEON experience to further push the limits & obtain a highly vectorized code before more of their competitors do.
- I.e. Developing their architecture to support ARM processors, team revamp to include SIMD SWE, putting out whitepapers, etc.