



2018 SOA Asia-Pacific Annual Symposium

Session 5B, Application of GPU in Actuarial Modeling

Presenters:

Joseph Kim, FSA, CERA, CFA, FCIA, FIAK

[SOA Antitrust Disclaimer](#)

[SOA Presentation Disclaimer](#)



The 8th SOA Asia Pacific Annual Symposium

25 May 2018



SOCIETY OF
ACTUARIES®

Application of GPU in Actuarial Modeling

JOSEPH KIM | FSA, FCIA, FIAK, CFA, CERA

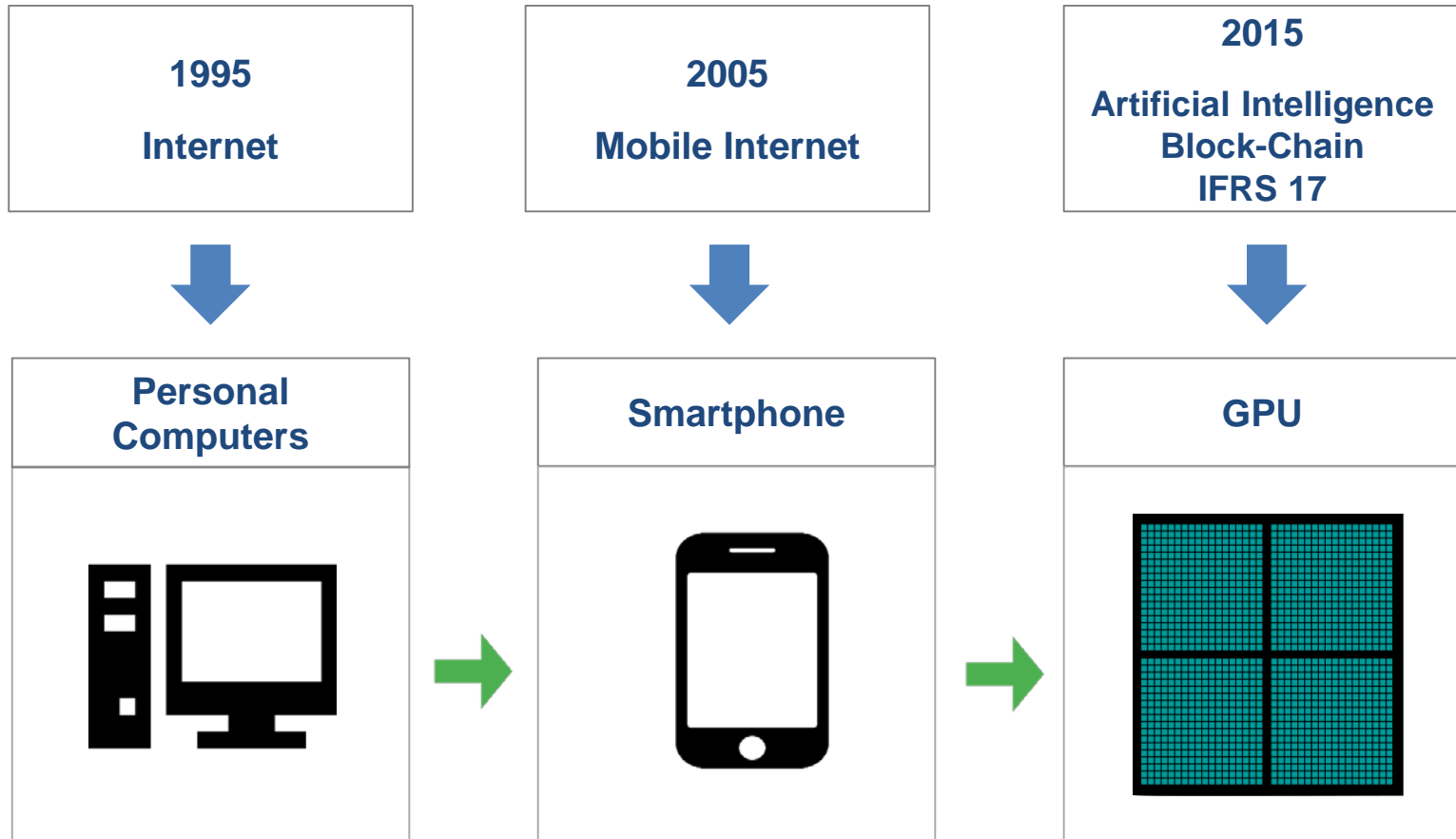
Consulting Actuary, Milliman

25th May 2018



Evolution of Hardware Technologies

Constant evolution in Information Technology is introducing new hardware to many people and industries.



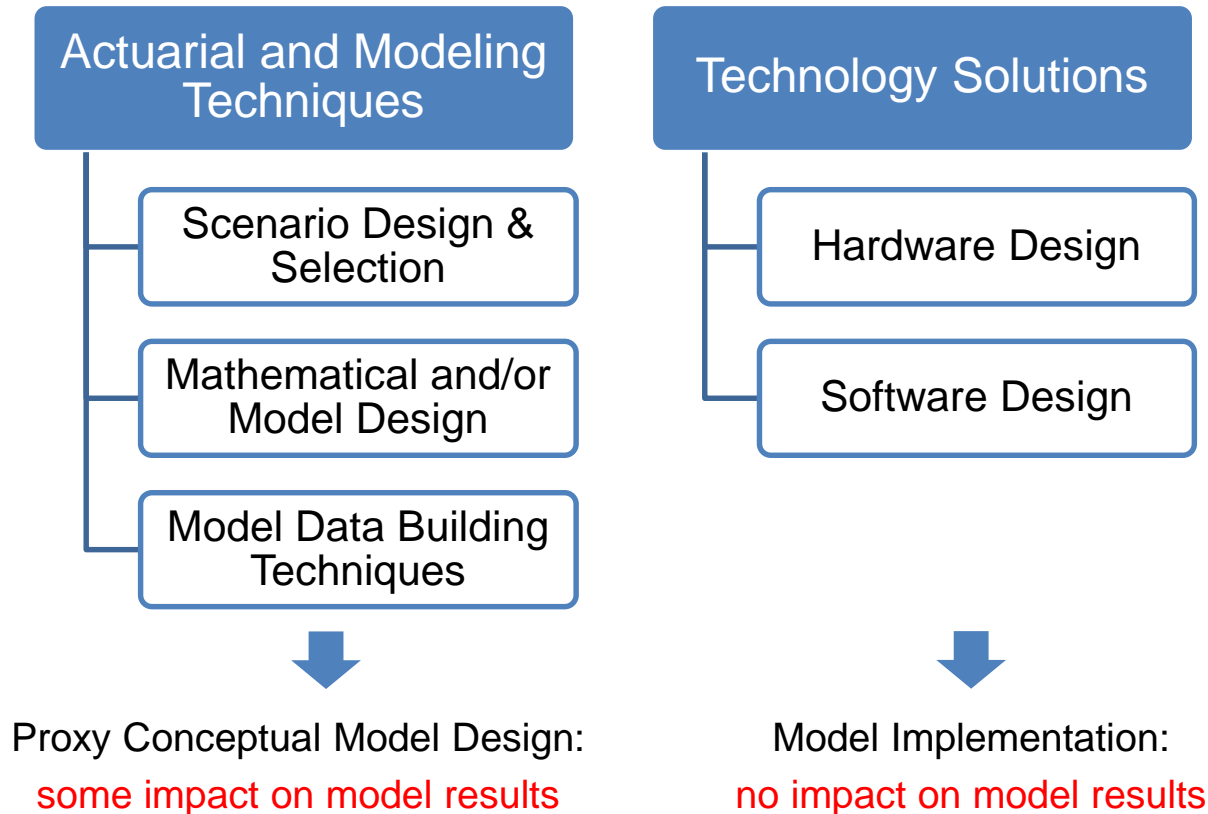
Challenges of IFRS 17 for Korean insurers

Adaptation of IFRS 17 implies a ground-breaking change for Korean companies' traditional financial reporting practice, significantly increasing levels for computing requirements.

Item	Changes	Increased Calc.
Reserve Principle	Net Level Premium (one period) → Gross Premium (all future periods)	x50
Unique Prod. Characteristics	Complex Products with many interdependent benefits, policy-holder options and choices of riders	Requires Seriatim Projection
Model Point	Clustered/Grouped (1%) → Seriatim (100%)	x100
Scenarios	Deterministic (1 Scenario BE or Worst Case) → Stochastic (average of 1000 scenarios)	X1000
MVMT	Need to isolate the impact from various changes 1 Run → 10 Runs	x10
Total	Significant increase in the computing power requirement	X 50,000,000

Model Efficiency Taxonomy

There are multiple ways to cope with the issue of increased computing requirements. Korean companies focus more on technological solutions such as GPU-computing.

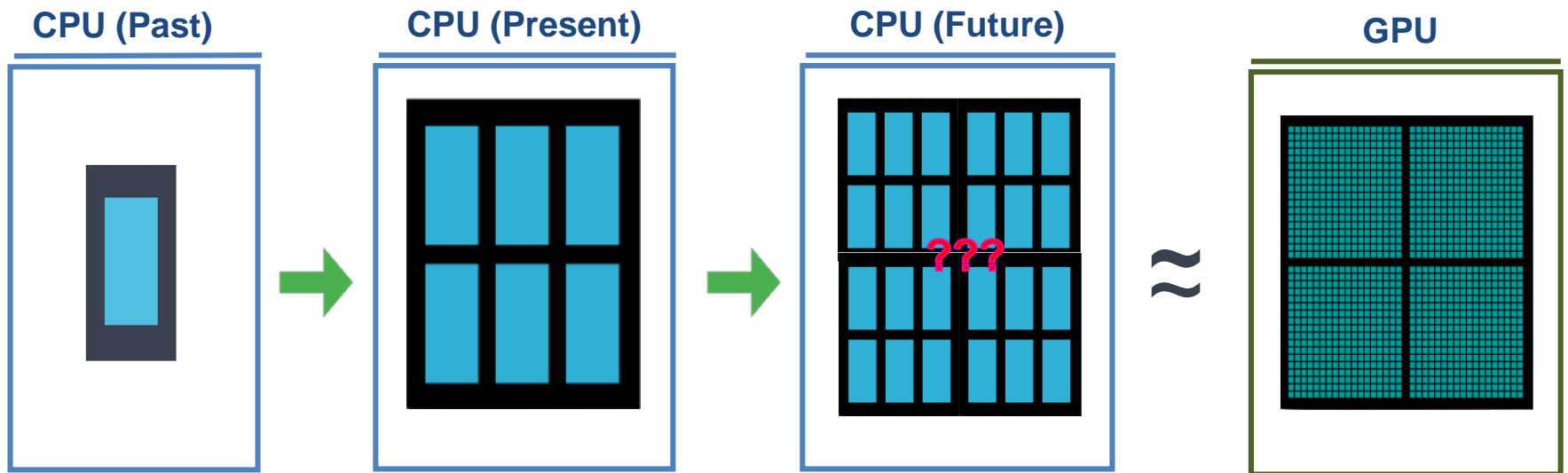


Source: 2016 SOA Life and Annuity Symposium
Session 57 PD: Model Efficiency - Part 1

Basics: CPU vs GPU

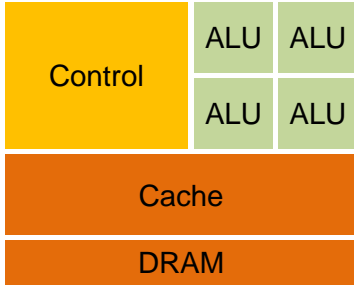
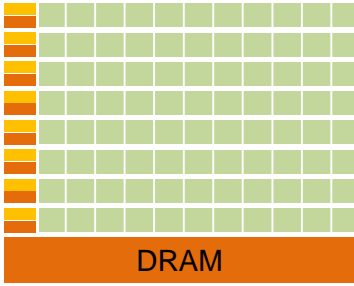
Original intension of GPU was to have a structural advantage over CPU in massive parallel processing to control many pixels of display devices.

- The capacity of a single-core has reached its limit for improvement (the end of Moore's Law: the number of transistors in a dense integrated circuit doubles about every two years)
- CPU makers then have been increasing the number of cores in a processor instead of increasing transistors in a core, introducing the multi-core technology.
- GPU has already had this multi-core structure from the birth, because it was originally developed to control many pixels of display devices.



More Detailed Comparison

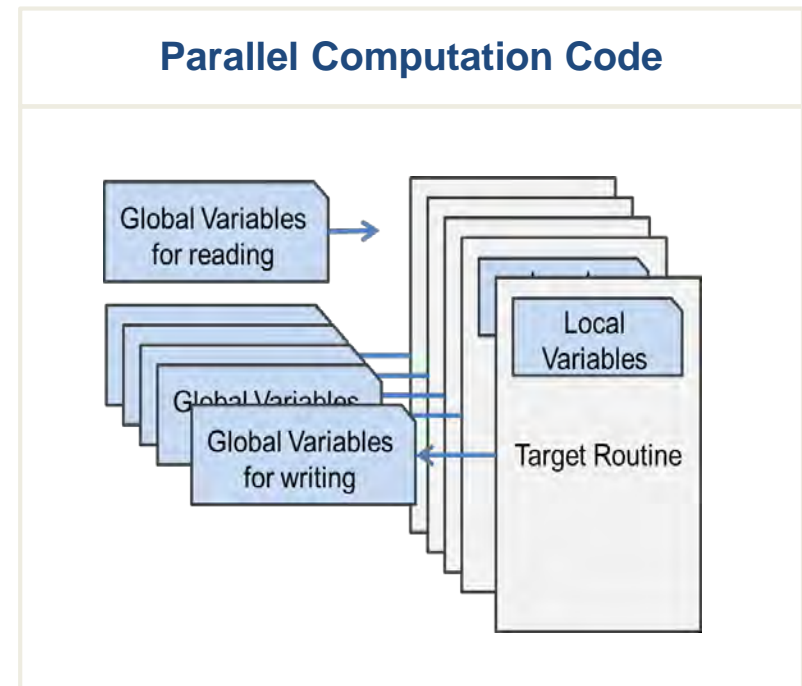
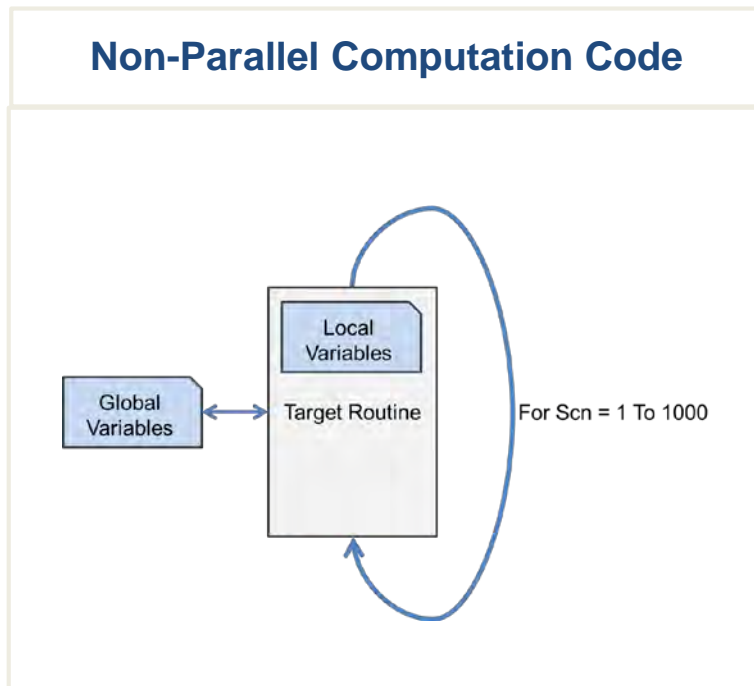
Both chips have very distinct pros and cons – the key is to apply where appropriate

	CPU	GPU
Structure	<ul style="list-style-type: none">A few number (4-28) of high-performing cores 	<ul style="list-style-type: none">Many numbers (5000+) of low-performing cores 
Strength	<ul style="list-style-type: none">Serial calculations (small number of complex order-dependent calc.)Relatively easy to programRelatively large size of memory per cores (efficient to handle large inputs)	<ul style="list-style-type: none">Parallel calculations (many numbers of short independent calculations)Less Expensive (Best per-Dollar Performance)
Weakness	<ul style="list-style-type: none">Relatively expensive to acquire thousands of cores	<ul style="list-style-type: none">Relatively difficult to use (because not developed for general purpose)Relatively small size of memory per cores (not efficient to handle large input data)

Limitations of GPU

Issue 1: Programming Difficulty

- Not practical for actuaries to learn and use CUDA C code for their daily modeling tasks. Both C and CUDA are difficult languages for even for most of seasoned programmers.
- **Solution:** GPU-based system should provide users with a easier DSL (domain-specific language) which gets translated into CUDA C codes.



Limitations of GPU

Issue 2: Limited size of memory

- Since GPU's memory has to be shared by 5000+ cores during parallel processing, the size of memory potentially allocated to each core is very limited.
- If the calculation of each core requires more than they are allocated, GPU cores would have to make data I/O to CPU's memory through PCI-Express channel which is going to be a lot slower than GPU's internal memory I/O.
- In this respect, GPU is most efficient to process one model point's multi-(inner)scenarios in parallel. However, the level of parallelization is then going to be limited and hence the performance enhancement through the parallelization is also going to be limited.
- To increase the level parallelization, more model-points or all model-points (for ALM)'s information would be required to be hold inside GPU cores. This would not be efficient/practical for GPU.

Logic Flow for Typical Actuarial Projection

```
Public Sub Main()
```

```
  Call Import_Global_Inputs
```

```
  For Outer_Scenario(ALM) = 1 to 1000
```

```
    For Policy = 1 To N
```

```
      Call Import_ModelPoint_And_Inputs
```

```
      For Inner_Scenario (Val'n) = 1 To 1000
```

```
        Call Projection (Calculation)
```

```
      Next Inner_Scenario
```

```
    Next Policy
```

```
  Next Outer_Scenario
```

```
  Call Export_Results
```

```
End Sub
```

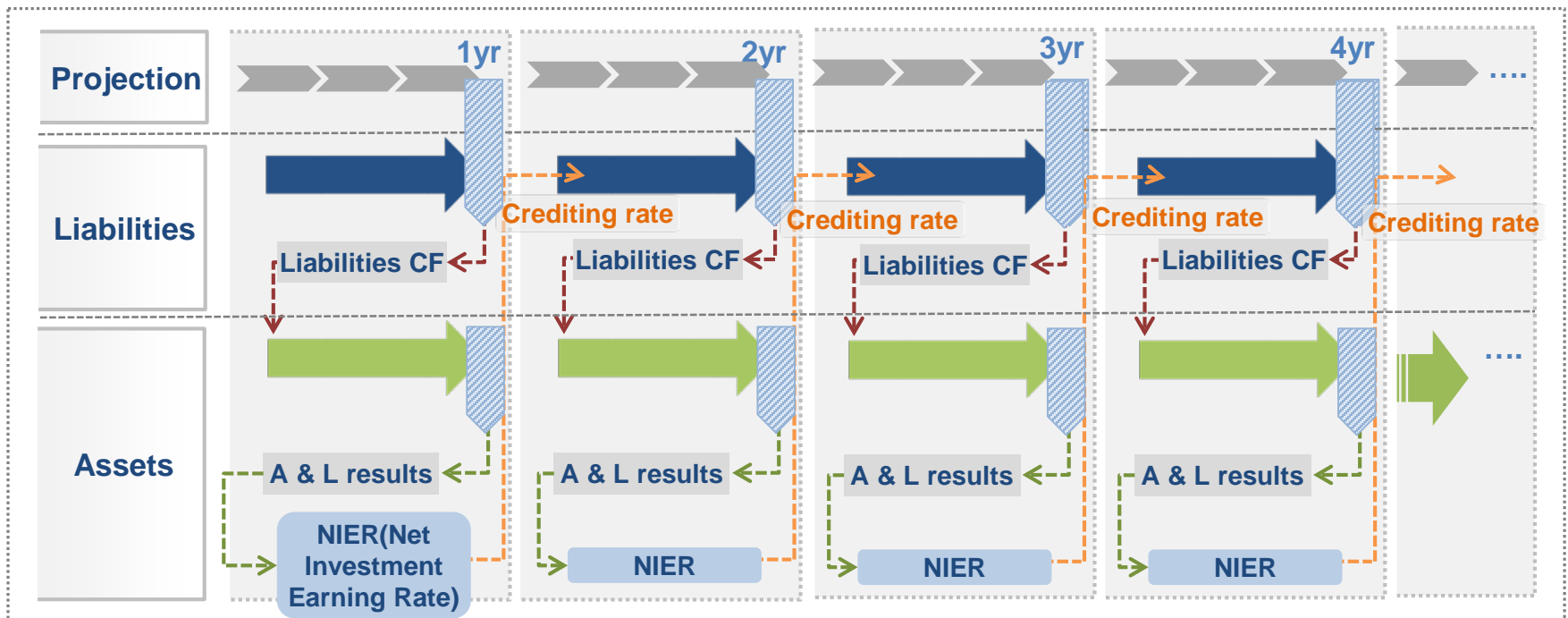
Limited
(Inner) Scen Loop
Parallelization

Higher-Level of
Parallelization

Limitations of GPU

Issue 2: Limited size of memory – dynamic ALM projection

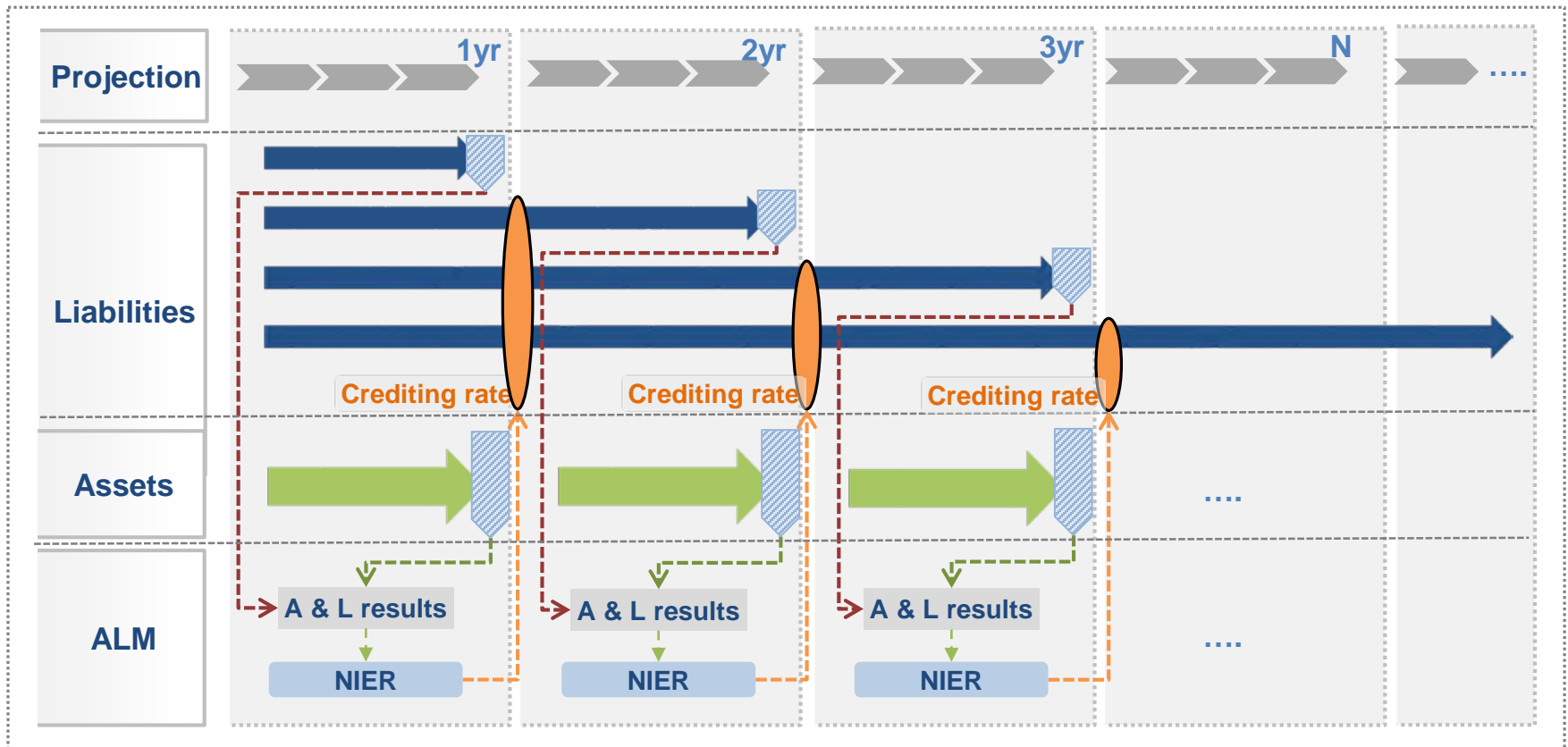
- Some insurance contracts require dynamic crediting rate mechanism to appropriately project their cash flows. In this case, all liability model-points and asset model-points have to be held in memory and processed together until the end of projection, unlike a typical liability-only projection where each model-point can be processed one after another.
- This type of projection produces a huge memory bottle-neck issue even with CPU.



Limitations of GPU

Issue 2: Limited size of memory – dynamic ALM projection

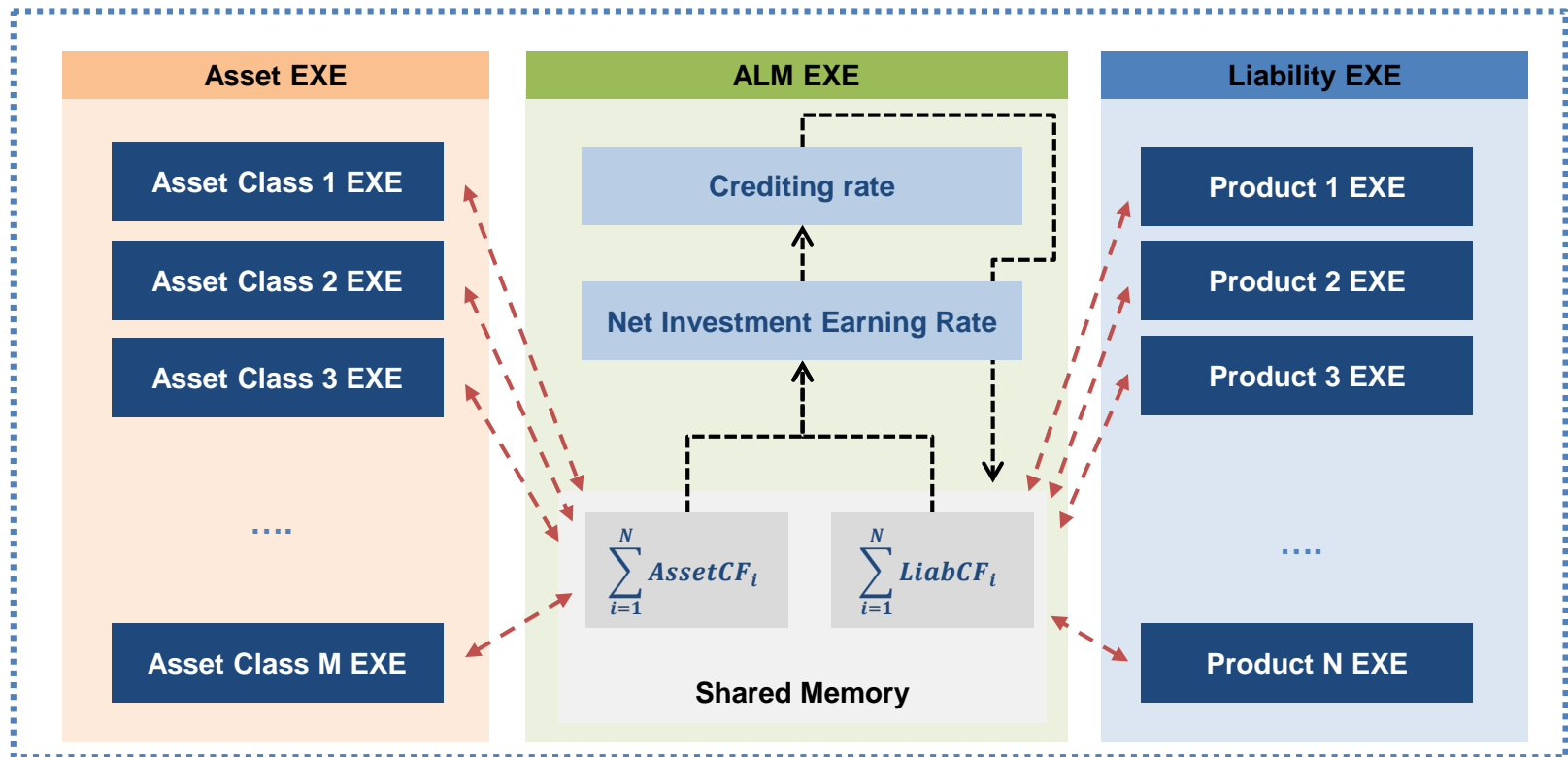
- Solution 1 (Traditional Approach): Use clustered model points (millions of MPs into hundreds)
- Solution 2 (Memory-Free Approach): T0 to Tx recalculation (not efficient speed performance)



Limitations of GPU

Issue 2: Limited size of memory – dynamic ALM projection

- Complementary Solution (Shared-Memory Approach): Parallel computations may involve multiple execution files. Hence, they require shared-memory technology to make independent execution files to communicate with each others – just like one execution file.



Limitations of GPU

Issue 3: GPU's massive core structure is not the best suited for complex sequential logics

- Complex actuarial logic may slow down the performance of GPU significantly.
- Due to the native structure with massive cores, all of GPU cores have to process the same command at the same time and cannot process different logics like CPU cores.
- This does not fit well with typical actuarial calculation logics with a lot of "IF" and "ELSE IF" statements

1. Simple Case (CPU vs GPU)

	CPU 1	...	CPU 4	GPU1	...	GPU1000
Do 1	Process		Process	Process		Process
Number of total processes	1	1	1	1	1	1

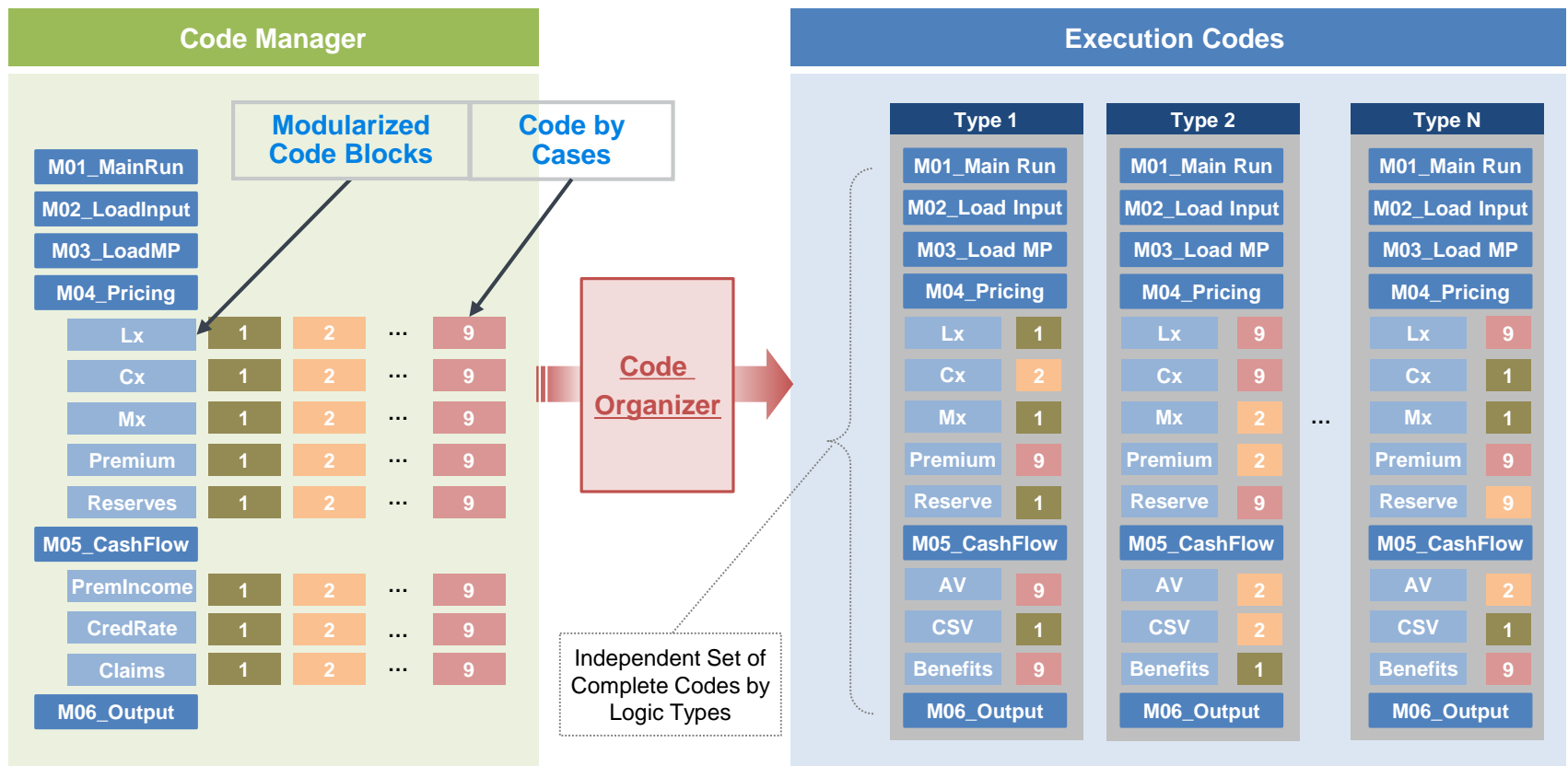
2. Typical Actuarial Calculation (Complex Condition Checks)

				CPU 1	...	CPU 4	GPU1	...	GPU1000
				A=TRUE		A=FALSE	A=TRUE		A=FALSE
				B=TRUE		C=FALSE	B=TRUE		C=FALSE
IF Condition A = True				Process		Process	Process		Process
	THEN	IF Condition B = TRUE		Process			Process		Wait
			THEN	Do 1			Process		Wait
			ELSE THEN	Do 2			Wait		Wait
	ELSE THEN	IF Condition C = TRUE				Process	Wait		Process
			THEN	Do 3			Wait		Wait
			ELSE THEN	Do 4		Process	Wait		Process
Number of total processes				3	3	3	7	7	7

Limitations of GPU

Issue 3: GPU's massive core structure is not the best suited for complex sequential logics

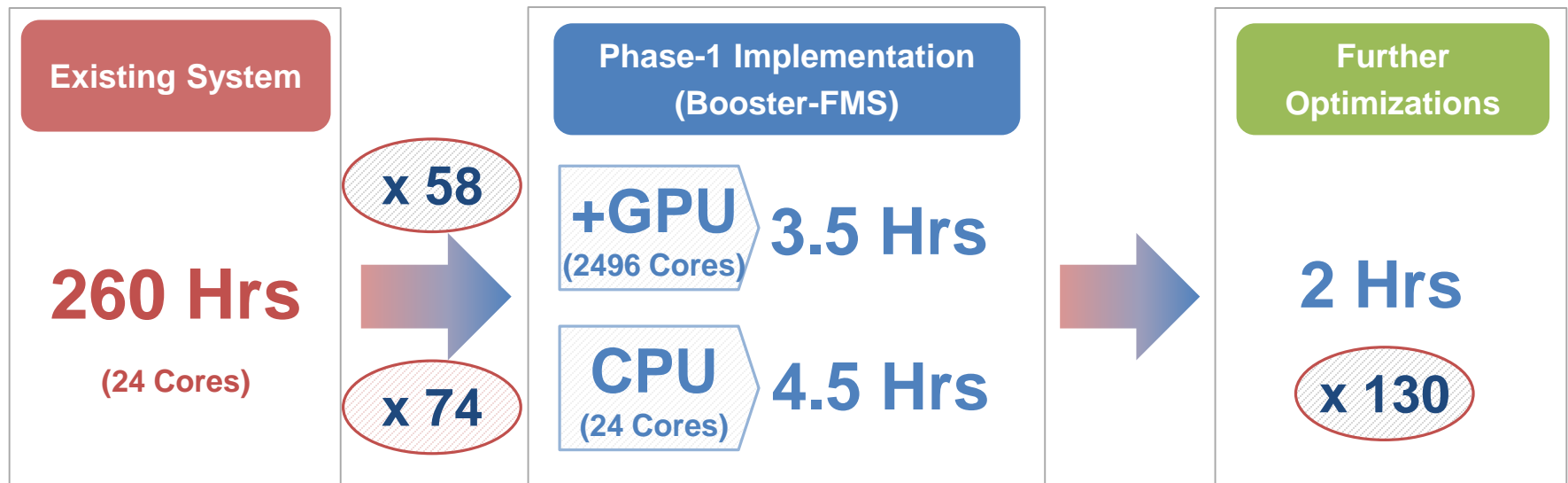
- **Solution:** GPU-based solution should provide advanced modular code management features which can populate many sets of simple/efficient codes (with less conditions) instead of a single set of complex codes (with a lot of conditions)



Actual Implementation Result

Efficient codes run fast regardless of the computing environment whether it is CPU or GPU.

- We implemented one client's GMxB reserve model based on Milliman's GPU-acceleratable solution, Booster-FMS.
- After the migration, we were able to reduce the run-time significantly from 260 hours to 3.5 hours based on the same server with an additional GPU card. However, the model has almost the same performance improvement even without utilizing GPU.
- Considering the scalability of the hardware, GPU may still the best per-dollar performance. However, CPU may still be fast enough if the code/data is well-optimized.



Key Takeaways

GPU has advantages in massive parallel computation and may achieve the best per-dollar performance but also has many limitations and requires more expertise than using CPU.

- GPU's unique structure with so many small cores make it superior for massive parallel computations. However, it also has many limitations: low usability, limited memory size and etc.
- CPU can perform as great as GPU if codes are well optimized. Otherwise, it will be 100 times slower and then the only viable solution with the speed would be proxy techniques.
- Technology will continue to evolve
 - ❖ GPU will try to be more like CPU: larger memory and faster data I/O
 - ❖ CPU will try to be more like GPU: more cores
- Who is going to be the winner?
 - ❖ Not sure yet. However, it's sure that both will improve at a rapid pace and they are going to affect the modeling approaches from proxy technique to more principle approaches. The seriatim / stochastic / nested-stochastic / dynamic ALM interaction will be the norm, soon.

Contacts

Joseph Kim, FSA, FCIA, FIAK, CFA, CERA

Consulting Actuary – Seoul office

Joseph.Kim@Milliman.com

+82-10+3172+3639

