TopQuants Autumn Workshop 2014
KPMG, Amstelveen, November 12, 2014

HPC for valuation of complex insurance guarantees

Jok Tang, Koos Huijssen (VORtech)
Denys Semagin (NN Group, ING Re)

VORTECH

NN

Disclaimer

From both ING Re and VORtech.

The contents provided in this presentation are informative in nature only, and do not express any official position or statement of either ING Re or VORtech. Both ING Re and VORtech take no responsibility for inaccuracies or interpretation of the information.

Outline

Part I

- ING Re introduction
- Complex insurance guarantees: variable annuities
- Valuation models for variable annuities and computational challenges

Part II

- VORtech introduction
- High-performance computing for valuation of variable annuities
- Scenarios for implementation

Outline

Part I

- ING Re introduction
- Complex insurance guarantees: variable annuities
- Valuation models for variable annuities and computational challenges

Part II

- VORtech introduction
- High-performance computing for valuation of variable annuities
- Scenarios for implementation

NN Group and ING Re introduction

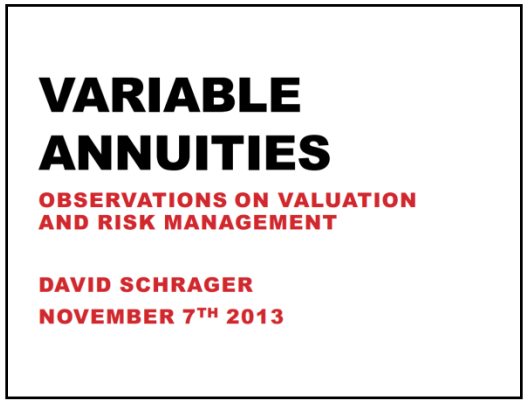
NN Group	An insurance and investment management company active in more than 18 countries, with a strong presence in a number of European countries and Japan. Formerly part of ING Group, NN Group listed as an independent company on Euronext Amsterdam on 2 July 2014. See more at www.nn-group.com
What is ING Re	The NN Group Reinsurance and Hedging Company • Internal NN Group reinsurer • Owner of the hedging program for VA Japan and VA Europe
ING Re's purpose	Achieve capital benefits by reinsurance and hedge risks with the market • Offer reinsurance solutions for NN Group business units • As an internal reinsurer, free up capital for the shareholders • Maintain top quality hedging platform for NN Group • Work towards a well-diversified, transparent and healthy portfolio

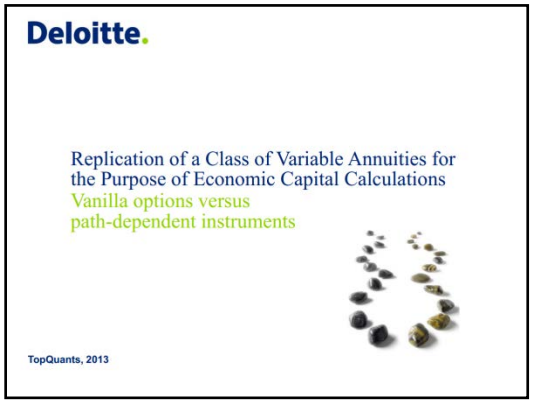
Variable Annuity (VA): What is it?

- Complex unit-linked (life) insurance guarantees.
- Insurer invests Customer's premia buying units of mutual funds
- Insurer guarantees the invested amount (less fees/costs)
- The (variable) guarantee is linked to death and other rider benefits.
- Embedded options with exposure to market and non-market risks.
- Variety of benefits, investment profiles, and holding periods: whole universe of challenges for pricing and risk management!

Challenges of VA valuation

- Business challenges of VA pricing and hedging have been actively discussed in recent years, including earlier talks at TopQuants:





- Additionally, it is a major modeling and computational challenge, hence ever growing practical needs for HPC as we discuss here.

Basic Variable Annuity: How it works?

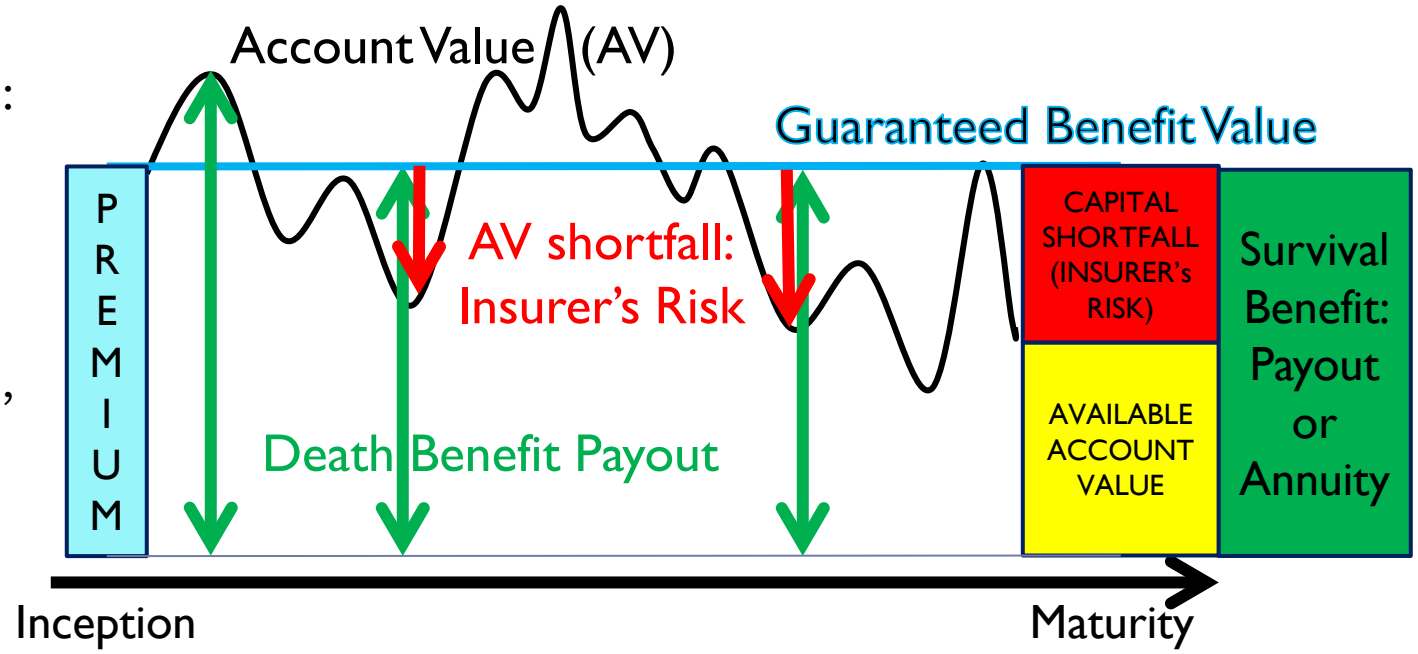
Customer chooses the amount, type of premium, and holding period. Insurer invests the assets: premium → funds → account value (AV).

Insurer offers death benefit (at any point) and survival benefit (at the end), and other benefits, composition of which defines Insurer's risks.

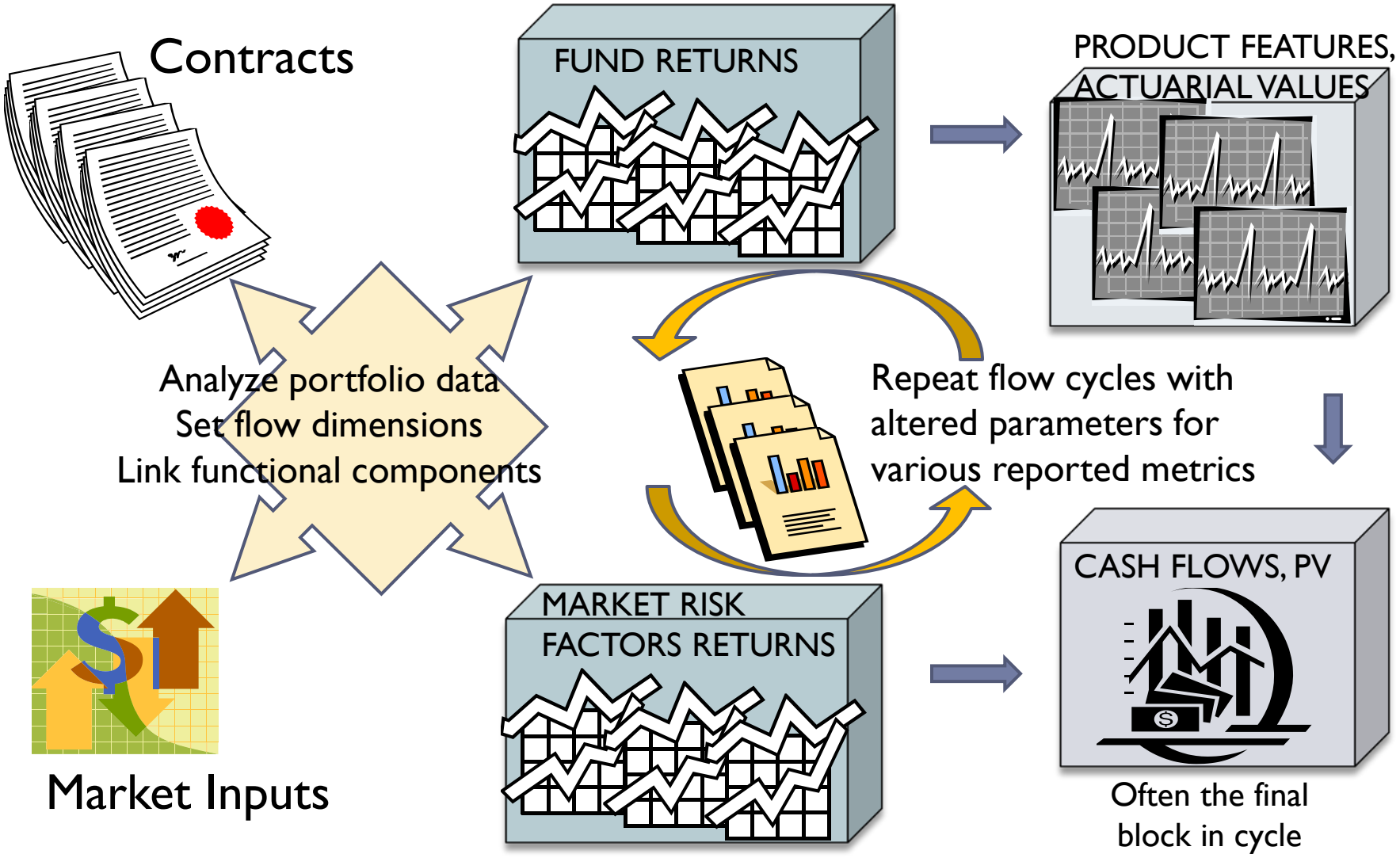
Customer bears running cost/fees, and can lapse the contract at any time and withdraw the available account value (American option).

Benefit payout: at least the guarantee G, or the AV if AV>G.

Risk: For AV<G, the (re-)insurer covers the AV shortfall.



General approach to modeling VAs



Liability model base (schematic)

Stochastic process for market risk factors

$$S_{\text{sc, t+1, rf}} = S_{\text{sc, t,rf}}\, r_{\text{sc,t}}\, dt + S_{\text{sc, t, rf}}\, \sigma_{\text{t,rf}}\, dz_{\text{sc,t,rf}}, \qquad dz \sim N(0, dt)$$

Account value as weighted combination of funds, regressed on risk factors

$$AV_{\text{sc,t}} = \sum_{f} w_f\, F_{\text{sc},t,f} = \sum_{f} w_f \sum_{\text{rf}} \beta_{f,\,\text{rf}}\, S_{\text{sc},t,\text{rf}}$$

Payout / cash flow projections for the liability guarantees per benefit type

$$CF_{\text{sc,t,b}} = \max\left(G_{\text{sc,t,b}} - AV_{\text{sc,t}}\,, 0\right)$$

Present value per benefit type as expectation of relevant discounted cash flows

$$PV_{\text{b}} = E_{\text{sc}}\left[\sum_{t} CF_{\text{sc},t,\text{b}} \exp\left(-r_{\text{sc},t}\; t\right)\right]$$

Model Dimensions and Complexity

Array/mesh of uniform random numbers, Nrn

Collection of Monte-Carlo projections, all compatible w.r.t. scenario and time dimensions

Nsc x Nt

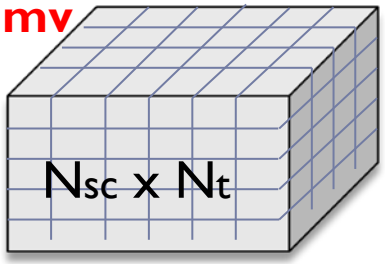
Present values per benefit and other derived metrics

PV_{b}

Nrn >= Nmv x Nsc x Nt

Nf >= Nb

Nmv



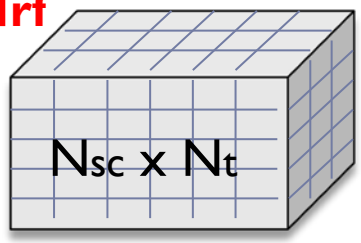
Multi-variates

$dz_{\mathrm{sc,t,mv}}$

Nmv >= Nrf

Nrf =< Nf

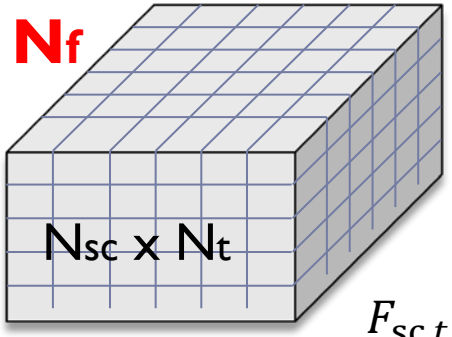
Nrf



$S_{\mathrm{sc,t,rf}}$

Risk Factors

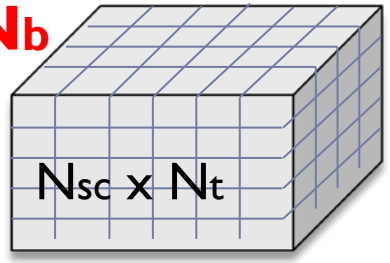
Nf



$F_{\mathrm{sc},t,f}$

Funds

Nb

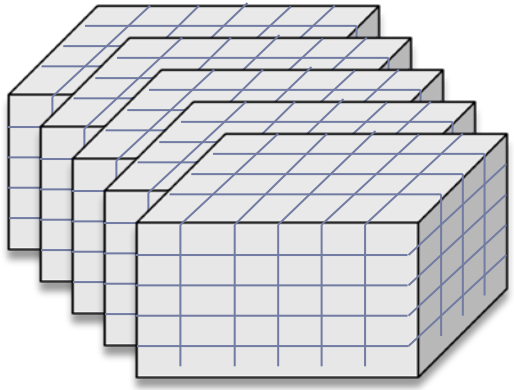


Benefit payout

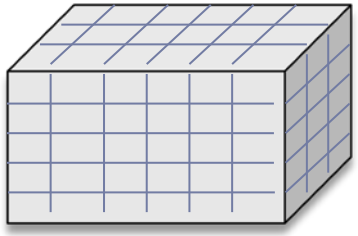
$CF_{\mathrm{sc,t,b}}$

Model Dimensions and Complexity - 2

Multiple cycles of per-policy processing



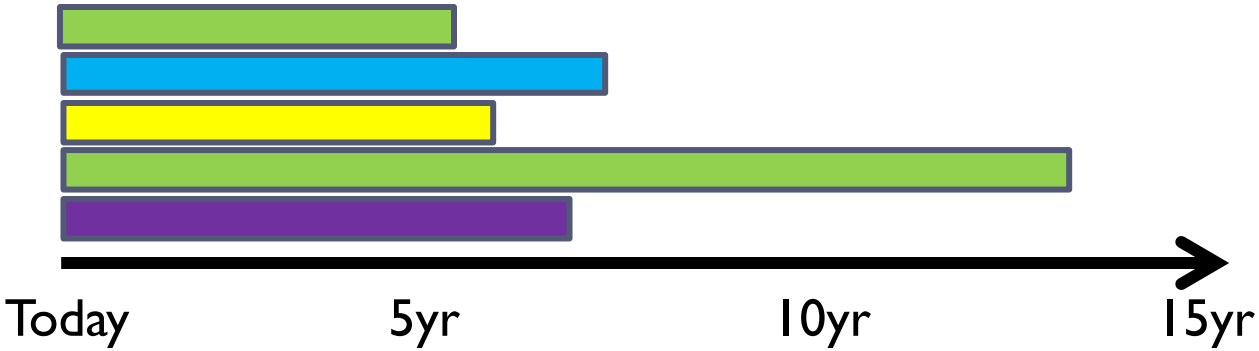
Target: Aligned collection of results projections (cash flow, pv) per scenario



Fund profile

A	B	C	D	E	F
✓		✓		✓	
			✓		
		✓			
				✓	✓
✓	✓	✓	✓	✓	✓

Product profile: **time to maturity**, guarantee levels, moneyness, surrender, fees, age, mortality rates, etc.



Model focus for HPC: functional flow

Seek balance between the scope of covered functionality and the amount of data shared across the following parallelized modules (focus on scalability and efficient memory management):

- Random number generation and relevant transforms
- Stochastic scenario projections for risk factors and funds
- Processing policies by type, risk profile and other criteria
- Path-dependent values of various actuarial/cashflow models
- Other manipulations of large cash flow arrays for various reporting.

Model focus for HPC: atomic algorithms

The estimate of target speed-up potential boils down to performance metrics for the representative assignment operations on large arrays (assuming efficient data/memory management):

- Simple referencing to other arrays/variables and using + , – , *

  ➔ $D = A*B - C;$

- More costly operations like division, exponent, log etc

  ➔ $D = \exp(A/B);$

- Mix of the above referencing to other parts of the same array

  ➔ $D_{[i]} = D_{[i-1]} * A_{[i]}/A_{[i-1]} ;$

- Conditional switch of the parameters and/or functional form

  ➔ $D_{[i]} = D_{[i-1]}/\exp(A_{[i]} - B_{[i]})$ if $B_{[i]}<A_{[i]}$; else $D_{[i]}=\alpha*D_{[i-1]}$;

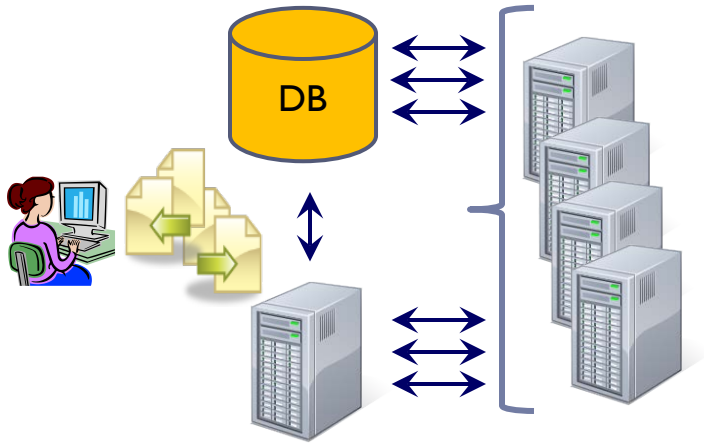
Production platforms for VA valuations

Purpose: risk management, trading, reporting, product development;

Focus: perfomance, robustness, user convenience, R&D capabilities;

One example: ING Insurance Library at ING Life Japan (e.g. presented at SoA conference, Tokyo 2010, http://www.soa.org/files/pd/2010-tokyo-ebig-rallis.pdf)

- Performance-driven hedging platform;
- Scalable isolated instances of engine;
- Designed with strict security and robustness requirement;
- Some prototyping / ad-hoc research done by VBA replica (~100x slower per core)



Practical challenges running a platform

Hardware / Software	• Limitations of datacenter space and available hardware • Historically chosen specific design and parallelization approach
Business Dynamics vs Resources	• Resource utilization patterns for ever changing contract profiles • Uneven workload on flow modules for different business needs • Changing priorities and allocation of dedicated resources
Operations / Continuity	• Trained staff with advanced programming and business skills • Prototyping capabilities, code transparency and flexibility • Respond to growing business needs for more granular insights • Respond to developing technology and regulatory environment

High level HPC needs for VA revisited

- Adopt new affordable power and technology options for performance
- Optimize the HW costs and dependencies with new trends in hosting
- Address legacy design limitations (e.g. memory, parallelization)
- Adapt to more commonly available skillset for active contribution
- Retain more prototyping flexibility and transparency of the solution

As an option:

Take a step back to prototyping environment (e.g. VBA) and consider a design focusing on productivity, flexibility, transparency, and costs

Outline

Part I

- ING Re introduction
- Complex insurance guarantees: variable annuities
- Valuation models for variable annuities and computational challenges

Part II

- VORtech introduction
- High-performance computing for valuation of variable annuities
- Scenarios for implementation

VORtech introduction

Company overview

- Established in 1996.
  Spin-off from Delft University of Technology.
- Office located in Delft.
- Organization growing gradually.
  Over 20 employees in 2014.
- Highly qualified employees.
  Over 50% holds a PhD degree.
- Key expertise:
  - Scientific software engineering;
  - Mathematical consultancy.

VORtech introduction

Application domains

- Water management
- Traffic and infrastructure
- Climate and environment
- Energy and oil industry
- Process and plant industry
- Finance
- Education and research
- and more…

VORtech introduction

Business unit: VORfinance

- ***VORfinance focus:***
  - Develop solutions for financial-oriented problems
- ***Domain includes:***
  - Banks, insurance companies, risk-management departments, asset management firms
- ***Solution tools:***
  - Software engineering
  - High performance computing
  - Financial and actuarial mathematics
  - Numerical optimization

VORtech introduction

Products and services

- Enhance mathematical models and solution methods.
- Optimize software by high performance computing (OpenMP, MPI, CUDA/OpenCL).
- Professional implementation of financial scientific software.
- Organize courses and seminars

Variable annuity project

- VORtech is involved to:
  - Perform a model scan of the prototype code.
  - Provide suggestions for parallel system design based on prototype code.

- Some requirements:
  - **Performance requirement**
    code needs to be significantly faster;
  - **Flexibility requirement**
    flexibility of the code should be maintained;
  - **Transparency requirement**
    transparency of the code should be maintained.

Acceleration options

A few approaches of acceleration:

1) An HPC environment is adopted;
2) Parts of prototype code are ported to different programming environment;
3) Port prototype code as a whole to different programming environment.

In the model scan, profiling tests are done.

Goal: identify time-consuming components of prototype code.

Time-consuming parts of the code

- One specific time-consuming component, called hereafter **“MapRFs”:**

```
Loop over funds
     Loop over periods
          Loop over indices
               Loop over scenarios
                    Generate fund projections
               End loop
          End loop
     End loop
End loop
```

- Another time-consuming component, called hereafter **“CF Comp”:**

```
Loop over policies
     Loop over scenarios
          Loop over periods
               Loop over funds
                    Series of cash flow computations
               End loop
          End loop
     End loop
End loop
```

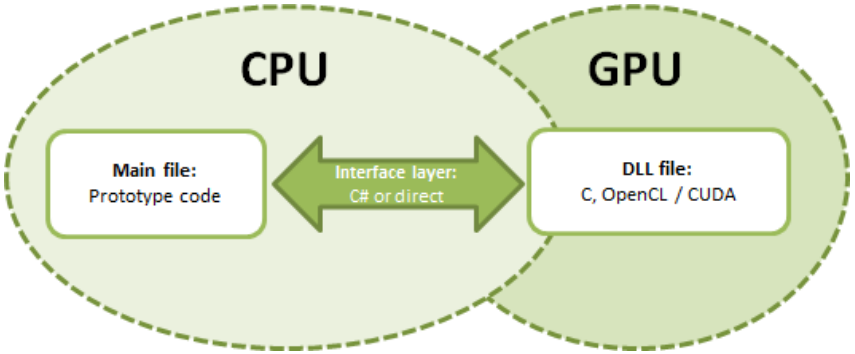
Solutions

Acceleration Option	GPU	Windows HPC server *	MATLAB (parallel computing toolbox)	.NET / C++ (CUDAfy, OpenMP, MPI)	Hybrid
CPU multicore computing	no	yes	yes	yes	yes
CPU cluster computing	no	yes	yes	yes	yes
GPU computing	yes	no	yes	yes	yes
Performance gain	very high	high	high	high	high
Hardware/software cost	low	moderate/high	high	moderate/high	moderate
Ratio gain / cost	high	moderate	moderate	moderate/high	moderate/high
Transparancy	low	high	moderate	moderate	low
Flexibility	moderate	high	high	moderate	moderate

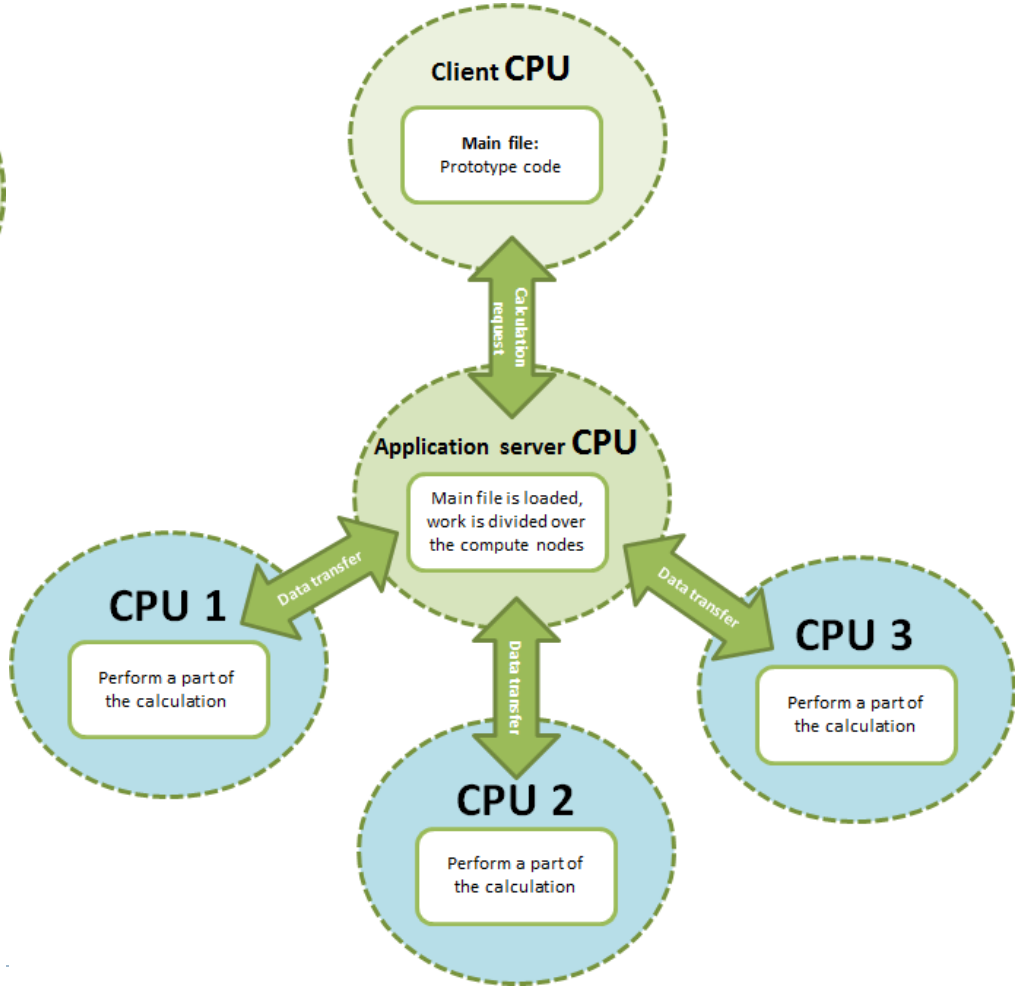
*Development cost of this option is low

High-Performance computing

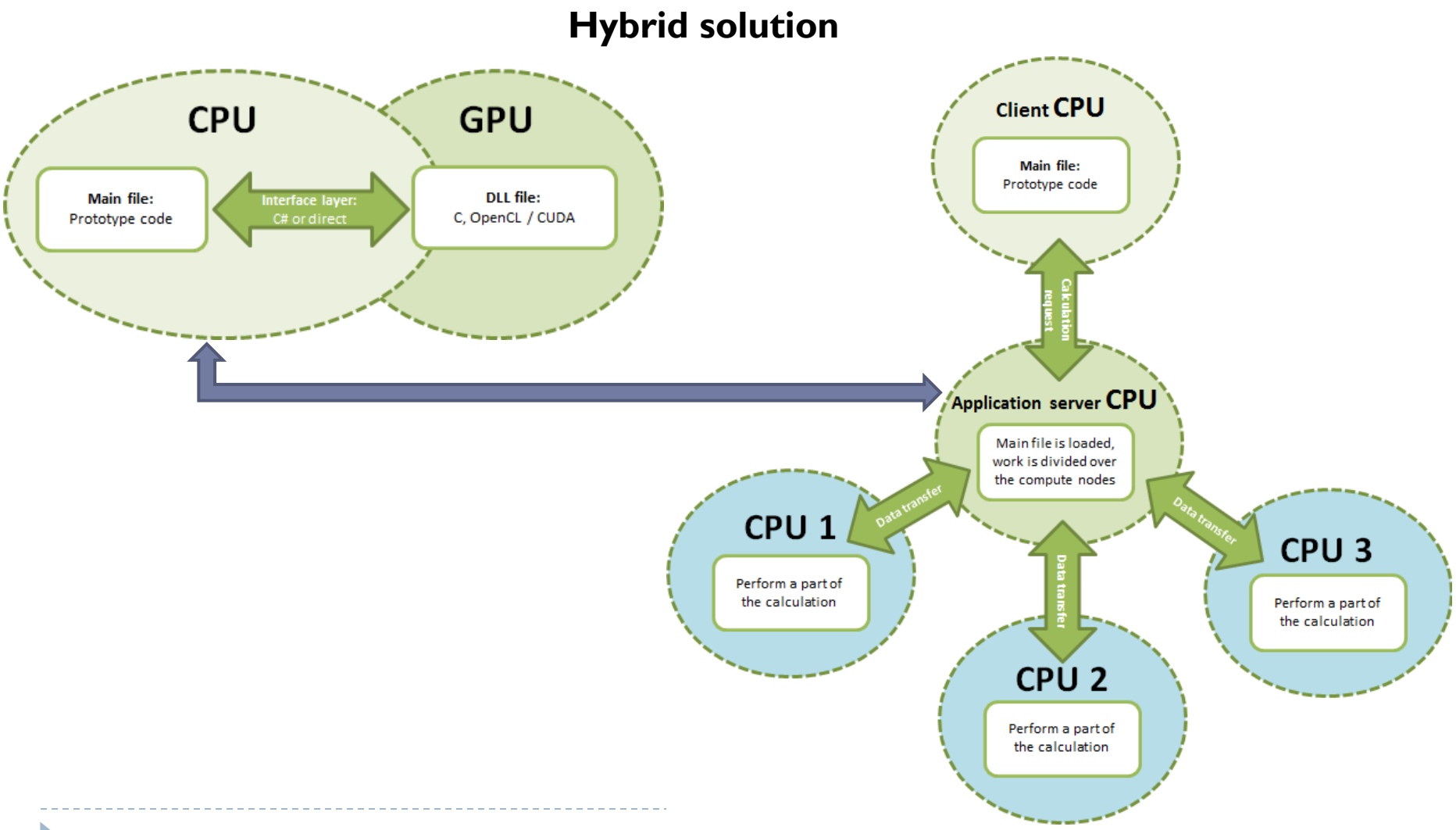
GPU solution



Windows HPC Server solution



High-Performance computing

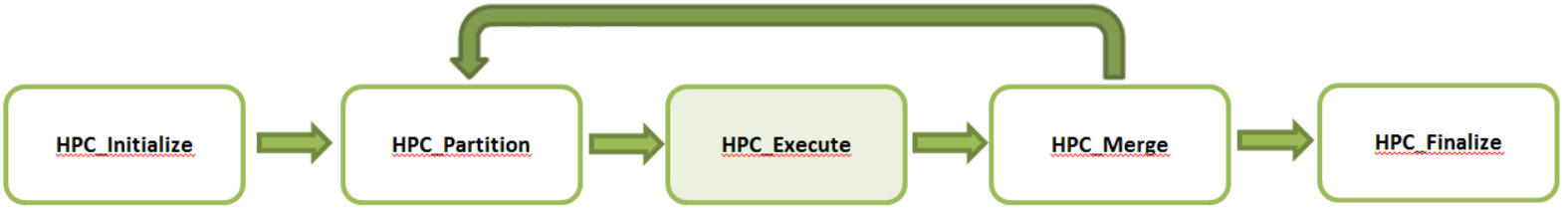


Windows HPC Server

- To support HPC Services for different prototype codes:
  current prototype code needs to include set of macros that implement asynchronous functions.

- Most important macros including their function names:

  - **HPC_Initialize**: perform any pre-calculation or initialization steps
  - **HPC_Partition**: collect required parameters for a single calculation step
  - **HPC_Execute**: perform one calculation step
  - **HPC_Merge**: process the results of a single calculation step
  - **HPC_Finalize**: perform any post-calculation processing



GPU Solution

In DLL (CPU/GPU):
* **Function:** Initialize various static data in host
* **Function:** Initialize device with static data
* **Function:** Draw random numbers
* **Different functions regarding:** Perform risk factors projection
* **Different functions regarding:** Perform risk factors to fund mapping ("MapRFs")
* **Different Functions regarding:** Perform cash flow computations ("CF Comp")
* **Function:** Calculate and send back output data

Separate DLL calls:

- Atomic kernels
- Multiple-entry points
- **Pro:** Flexible and transparent
- **Con:** Not fully efficient

A full DLL call:

- Holistic, single kernel
- Single-entry point
- **Con:** Not fully flexible and transparent
- **Pro:** Efficient

GPU Solution

- Tentative results for the test case **"CF Comp"**

	Test Problem			
Parameters	**1**	**2**	**3**	**4**
# Policies	100	100	1000	1000
# Scenarios	100	1000	100	1000
# Total Funds	48	48	48	48
# Active Funds	27	27	27	27
# Time Steps	191	191	191	191

	Computational time in sec			
Programming language	**1**	**2**	**3**	**4**
VBA (sequential)	5.2	55.0	49.9	533.1
C (sequential)	0.2	1.4	1.5	14.1
OpenCL (parallel)	0.02	0.02	0.02	0.15

Speedup factors	**1**	**2**	**3**	**4**
VBA to C	26.0	39.3	33.3	37.8
VBA to OpenCL	346.7	2619.0	2935.3	3554.0
C to OpenCL	13.3	66.7	88.2	94.0

C Computations:
- i7-2640, 2.80 Ghz
- 8GB RAM
- Double precision
- O2 optimization

OpenCL Computations:
- AMD Radeon HD 9870M
- 1280 cores
- Double precision
- "Standard" implementation

CPU-GPU Data transfer:
- GPU memory bandwidth 154 GB/s
- PCIe 3.0 bus bandwidth 16 GB/s
- Data transfer time for the Test 4 (~200MB) plus latency: ~10 ms.

GPU Solution

- Tentative results for different variants of **"CF Comp":**

Code	Test Cases	1	2	3	4
	# Policies	100	100	1000	1000
	# Scenarios	100	1000	100	1000
VBA (sequential)	**CF Comp (s)**	5.2	55.0	49.9	533.1
	CF Mod Exp (s)	10.9	111.9	104.0	1122.5
	CF Mod Base (s)	6.6	67.0	61.6	673.6
	CF Mod Cond (s)	9.4	97.3	92.5	960.2
C (sequential)	**CF Comp (s)**	0.2	1.4	1.5	14.1
	CF Mod Exp (s)	0.4	4.3	4.4	43.3
	CF Mod Base (s)	0.4	3.6	3.6	35.8
	CF Mod Cond (s)	0.4	3.6	3.6	35.6
OpenCL (parallel)	**CF Comp (s)**	0.02	0.02	0.02	0.2
	CF Mod Exp (s)	0.03	0.13	0.09	0.9
	CF Mod Base (s)	0.01	0.1	0.1	0.8
	CF Mod Cond (s)	0.02	0.1	0.1	1.0

CF Comp: standard.

CF Mod Exp: formula with an exponent.

CF Mod Base: formula with shifted array references.

CF Mod Cond: formula with a conditional switch of functional statements.

GPU Solution

- Tentative results for different variants of **"CF Comp":**

VBA -> OpenCL Speedup factors	1	2	3	4
CF Comp	346.7	2619.0	2935.3	3554.0
CF Mod Exp	330.3	895.2	1155.6	1261.2
CF Mod Base	471.4	609.1	733.3	863.6
CF Mod Cond	587.5	810.8	840.9	950.7

C -> OpenCL Speedup factors	1	2	3	4
CF Comp	13.3	66.7	88.2	94.0
CF Mod Exp	12.1	34.4	48.9	48.7
CF Mod Base	28.6	32.7	42.9	45.9
CF Mod Cond	25.0	30.0	32.7	35.2

CF Comp: standard.

CF Mod Exp: formula with an exponent.

CF Mod Base: formula with shifted array references.

CF Mod Cond: formula with a conditional switch of functional statements.

Main conclusions of the model scan

- Parallelizing prototype code is an adequate way to proceed.
- The two best acceleration options:
  - **GPU solution**:
    - will require a significant development effort,
    - investment of hardware is low,
    - flexibility and transparency will suffer.
  - **HPC server solution**:
    - requires a significant investment in a cluster,
    - development cost is relatively low,
    - flexibility and transparency will be maintained.

Possible scenarios for implementation

Acceleration Option	GPU solution	Windows HPC server solution
Explanation	Parts of prototype code will be outsourced to GPUs	Prototype code will run in HPC Server environment
Hardware requirements **Estimated hardware/ software cost ***	Machine including powerful GPU with 2000+ cores 1 unit	HPC cluster with at least 256 nodes 20 units
Software development **Development cost ***	Build DLL with CPU/GPU routines Build interface layer (in C#) Adapt original prototype code 7 units	Build HPC functions to prototype Adapt original prototype code 1 unit
Performance gain **Code transparancy** **Software flexibility**	High Low Moderate	High High High

* Units per row correspond to a different actual cost.

Thanks for your attention!

Contact information

Dr.ir. J.M. (Jok) Tang
VORtech
tang@vortech.nl
+3115 – 251 19 49

Denys Semagin, PhD
ING Re
Denys.Semagin@nn-group.com
+31 70 379 11 64