Accelerating Market Value-at-Risk Estimation on GPUs

NVIDIA Theater, SC'09

Matthew Dixon[1] Jike Chong[2]

[1] Department of Computer Science, UC Davis

[2] Department of Electrical Engineering and Computer Science, UC Berkeley

Problem

- Value-at-Risk (VaR) is an essential metric used by financial institutions to estimate a portfolio's sensitivity to market risks
- Estimations with realistic scenarios require computationally expensive Monte Carlo based VaR (MC-VaR) approaches
- Difficult for domain experts to develop and test efficient risk management systems

Solution

- We propose algorithmic and implementation approaches to speedup MC-VaR computations by up to 130x
- This makes MC-VaR more computationally viable, facilitating more pervasive stress testing
- Domain experts can develop, test and run a MC-VaR engine on their workstations
- Enable risk reporting or trading activity to become 'live’

Outline

- Overview of market risk
- Introduction to VaR modeling
- Solution Approach
    1. Problem reformulation
    2. Module selection
    3. Implementation styling
- Results
- Summary

Software Architecture

Sobol quasi random number generation (RNG)

Quasi random vectors

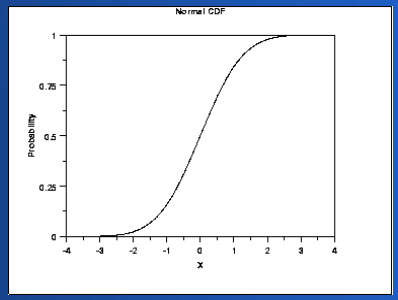
0, 0.5 ,0.25, 0.75, 0.375, 0.875, 0.125 …
0, 0.5 ,075, 0.25, 0.375, 0.875, 0.625 ...
...

Box Muller transformation /
Moro Inverse approximation

Normal quasi random vectors

-0.193736, 0, 0.0209739, -0.0209739, -0.00990839, 0.0357712 ...
-0.189, 0, -0.0204612, 0.0204612, -0.00966619, 0.0348968 …
...

Aggregation of the losses: linear algebra



Cumulative portfolio loss distribution

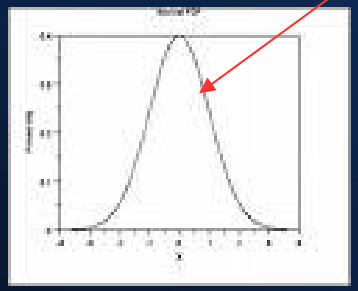
Quadratic VaR Estimation

Portfolio Loss

$$\Delta_i = \frac{\partial P}{\partial R_i} \qquad \Gamma_{ij} = \frac{\partial^2 P}{\partial R_i \partial R_j}$$

Portfolio Loss aggregation:

$$dP = \sum_{i,j} \underbrace{\Delta_i dR_i}_{delta} + \underbrace{\frac{1}{2} dR_i \Gamma_{ij} dR_j}_{gamma},$$



Assume joint distribution of risk factor returns
e.g. log-normal or student-t

Linear (delta) component

Non-linear (gamma) component

Our Solution Approach

- We use a variety of techniques to speed up MC-VaR computation

1. **Algorithmic reformulation:** Reorder the algorithm to avoid BLAS 3 routines
2. **Module Selection:** Select the random number generator with the fastest convergence rate in the standard error
3. **Implementation styling** Merge the random number generator and distribution transformation kernels

1. Algorithmic Formulation: Optimization

Standard aggregation:

$$\Delta P_k = \sum_{i,j} R_i \Delta_i Q_{ij} X_{jk} + \hat{\lambda}_i X_{ik}^2,$$

Eigenvalues of Gamma matrix

Portfolio Loss

Risk Factor

Delta

Matrix of correlated random numbers: matrix-matrix multiplication

Precompute q_j

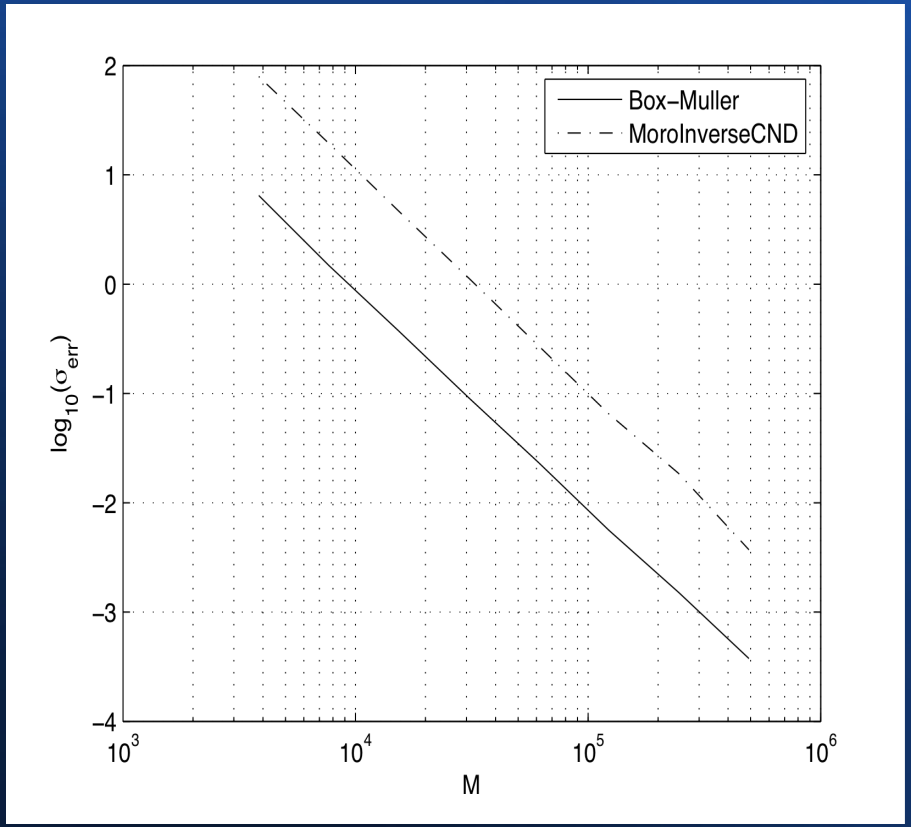
Optimized aggregation:

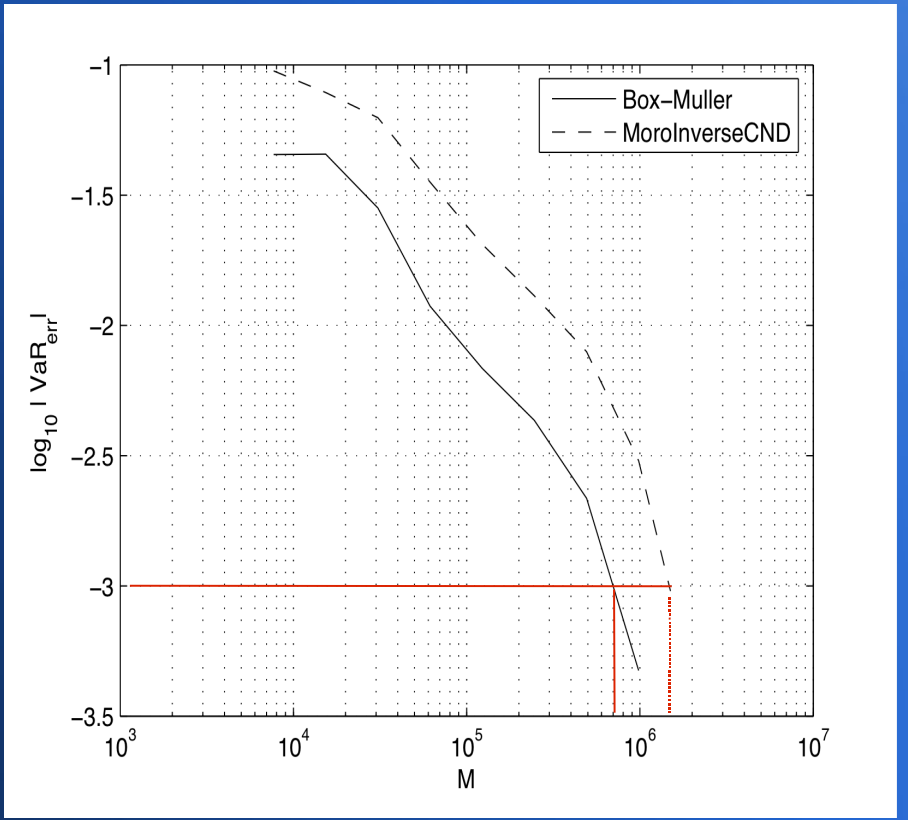
$$q_j := \sum_i R_i \Delta_i (c_i + Q_{ij}),$$

$$\Delta P_k(R_1, \ldots, R_N, t) = \sum_{i,j} q_j X_{jk} + \hat{\lambda}_i X_{ik}^2.$$

Vector of correlated random numbers: matrix-vector multiplication

2. Module Selection





(left) The standard error (%) in the portfolio loss distribution using Moro's interpolation method and the Box-Muller method applied to Sobol' sequences. (right) The error (%) in the simulated 1 day portfolio delta VaR (c=95%).

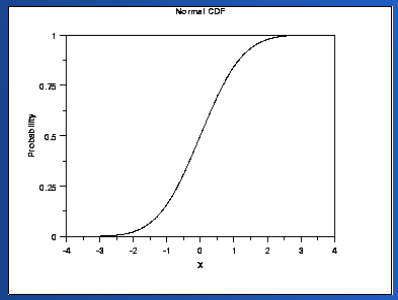
3. Implementation Styling

Merged
Sobol quasi random number generation (RNG)
and
Box Muller transformation

Normal quasi random vectors

-0.193736, 0, 0.0209739, -0.0209739, -0.00990839, 0.0357712 ...
-0.189, 0, -0.0204612, 0.0204612, -0.00966619, 0.0348968 …
...

Aggregation of the losses: linear algebra



Cumulative portfolio loss distribution

Overall Results

Δ – Γ (seconds)	URNG	Distribution Conversion	Aggregation	Total Time
Baseline CPU	9.08 (1.17%)	424 (54.5%)	345 (44.3%)	778
Baseline GPU	0.404 (0.375%)	0.429 (0.398%)	108 (99.2%)	109
Problem Formulation (GPU)	- (19.6%)	- (20.8%)	1.23 (59.6%)	2.06
Module Selection (GPU)	0.202 (19.7%)	0.211 (20.5%)	0.615 (59.8%)	1.03
Implementation Styling (GPU)	0.224 (26.7%)		- (73.3%)	0.839
Speedup (Compared to Baseline GPU)	**3.72x**		**176x**	**130x**

The overall speedup obtained by applying all three optimizations on the NVIDIA GeForce GTX 280. The baseline CPU is an Intel Core2 Q9300.

Algorithmic Reformulation

Δ − Γ	Standard	Reformulated	Speedup
CPU	173 (389)	0.551 (217)	314x (1.79x)
GPU	54.0 (54.4)	0.615 (1.03)	87.8x (52.8x)
Speedup	3.20x (7.15x)	0.896x (211x)	281x (378x)

Speedup from algorithmic reformulation of the loss estimation portion of the algorithm (with the overall timing in parentheses) on an Intel Core2 Q9300 and NVIDIA GeForce GTX280 using a portfolio of 4096 risk factors

Implementation Styling

	Standard (Separate)	Re-styled (Merged)	Speedup
Box-Muller (step 1)	0.299 sec	0.331	1.92 x
(step 2)	0.337 sec		
Bailey (step 1)	0.297	0.875	0.83 x
(step 2)	0.425		

Speedup from merging the Sobol' QRNG module with the Box-Muller or Bailey's module on a NVIDIA GeForce GTX280

Three Optimization Approaches

1. Reformulating the loss aggregation delivers a 50x speedup on the GPU
2. Generating normal variants from a Sobol' sequence is 2x more efficient using the Box-Muller method than the Moro approximation.
3. Merging the Sobol' sequence generator with the Box-Muller (Bailey's) method delivers up to 1.92x speedup.

Conclusions

- Domain experts need a MC-VaR engine that runs in seconds on their workstations
  - ✓ Enable more pervasive stress testing
  - ✓ Rapid application development
  - ✓ Comprehensive model validation
  - ✓ 'Live' risk analysis, reporting and control of trading
- We propose using the GPU to run MC-VaR at the client side instead of the server side

Overall Results

Δ – Γ (seconds)	URNG	Distribution Conversion	Aggregation	Total Time
Baseline CPU	9.08 (1.17%)	424 (54.5%)	345 (44.3%)	778
Baseline GPU	0.404 (0.375%)	0.429 (0.398%)	108 (99.2%)	109
Problem Formulation (GPU)	- (19.6%)	- (20.8%)	1.23 (59.6%)	2.06
Module Selection (GPU)	0.202 (19.7%)	0.211 (20.5%)	0.615 (59.8%)	1.03
Implementation Styling (GPU)	0.224 (26.7%)		- (73.3%)	0.839
Speedup (Compared to Baseline GPU)	**3.72x**		**176x**	**130x**

The overall speedup obtained by applying all three optimizations on the NVIDIA GeForce GTX 280. The baseline CPU is an Intel Core2 Q9300.

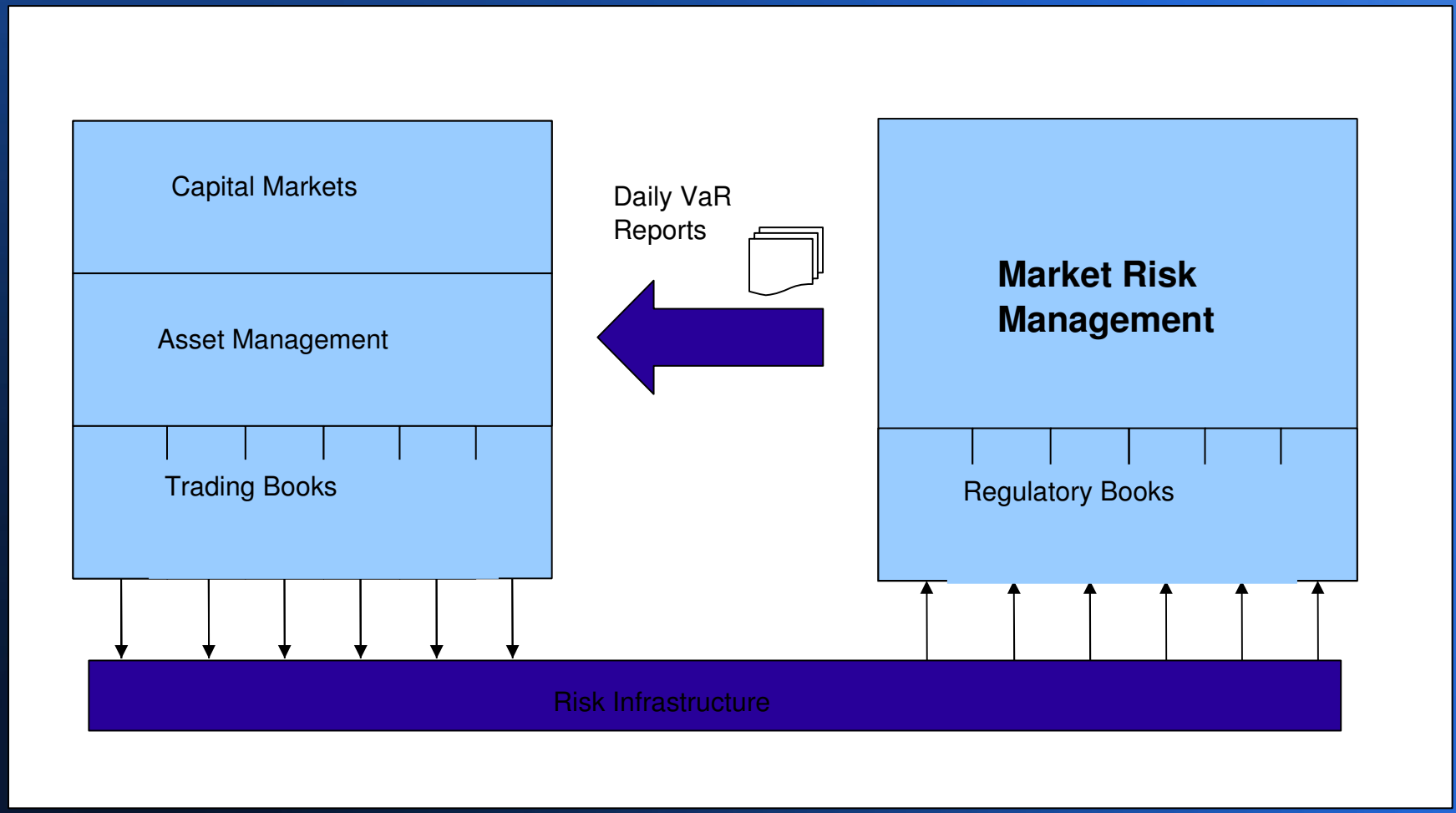
Financial Risk

- **Market Risk:** The market value of trading assets may fall.
- **Credit Risk:** Debtors (including market counterparties) may not be able to meet their obligations, causing a loss to the bank.
- **Operational risk:** Loss of money due to a failure of the bank's infrastructure or internal controls.
- **Liquidity risk:** Banks may incur costs in raising cash to meet obligations when fall due.
- **Business risk:** Business may become less profitable because of changes in the market for bank's services.

Market Risk

- The proliferation of algorithmic trading, derivative usage and highly leveraged hedge funds requires improved risk management systems
- Portfolios containing thousands to hundreds of thousands of tradable assets (or instruments) are not uncommon
- In day-to-day trading, financial institutions measure market risk with the Value-at-Risk (VaR) metric
- VaR estimates the potential severe falls in the market value of their portfolio over a short time horizon

Market Risk Advisory Mechanism



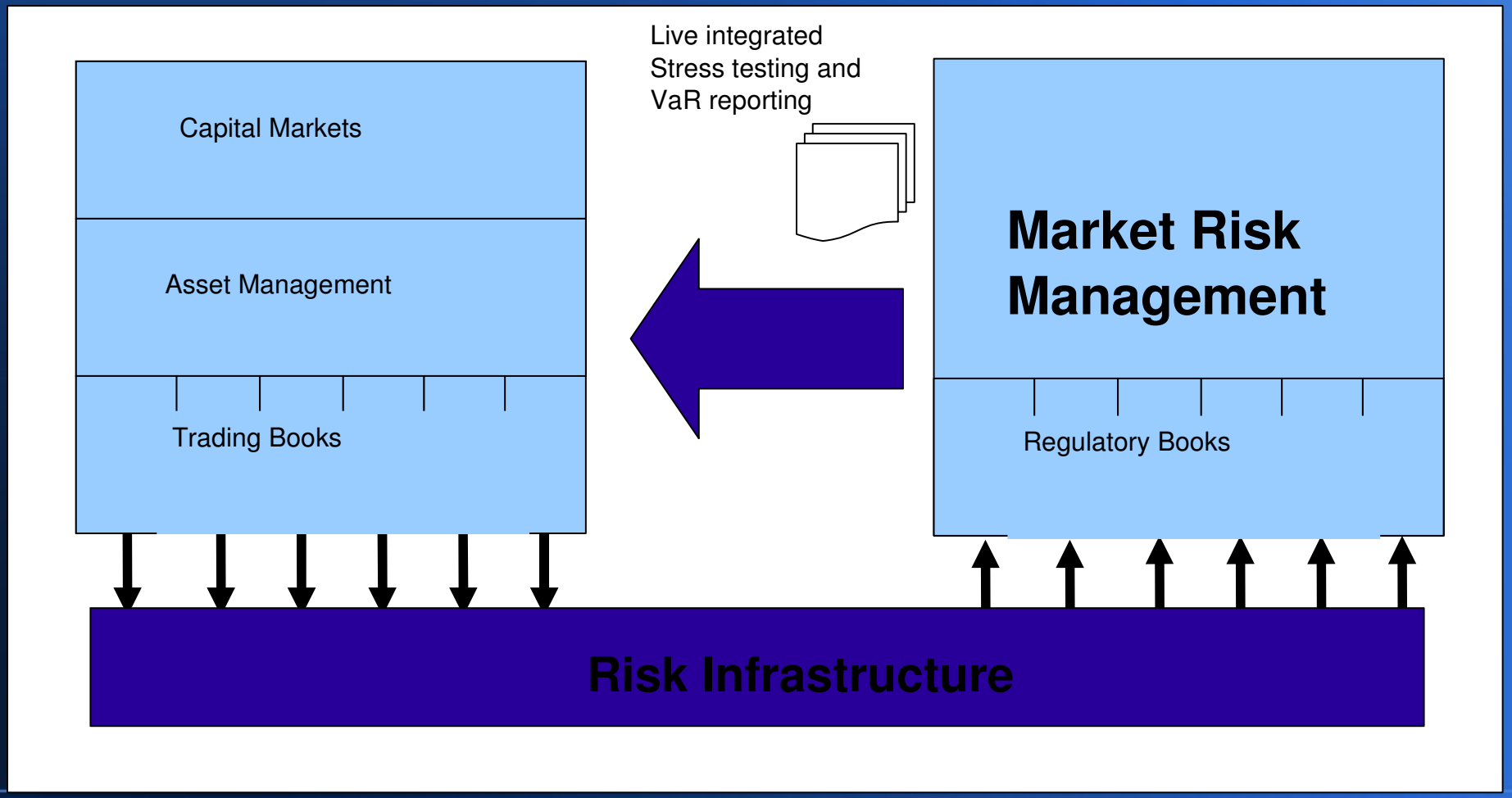
2009 BIS Stress Testing Guidelines

"At the most fundamental level, weaknesses in infrastructure limited the ability of banks to identify and aggregate exposures across the bank. This weakness limits the effectiveness of risk management tools - including stress testing ..."

"...Further investments in IT infrastructure may be necessary to enhance the availability and granularity of risk information that will enable timely analysis and assessment of the impact of new stress scenarios designed to address a rapidly changing environment"

Principles for sound stress testing practices and supervision - consultative paper, January 2009, Bank for International Settlements.

Required Market Risk Advisory Mechanism



Who needs fast VaR?

- *Market risk management operations* provide daily VaR reports, summarizing where individual business units are most vulnerable to market movements
- VaR is also frequently estimated on an ad-hoc basis by *business analysts*
- *Capital allocation managers* use VaR to optimize portfolios
- *Quantitative strategists* will devise trading strategies using VaR
- *Trade structures* will assess the impact of new bespoke structured products on their business unit

Why is VaR a challenge for developers ?

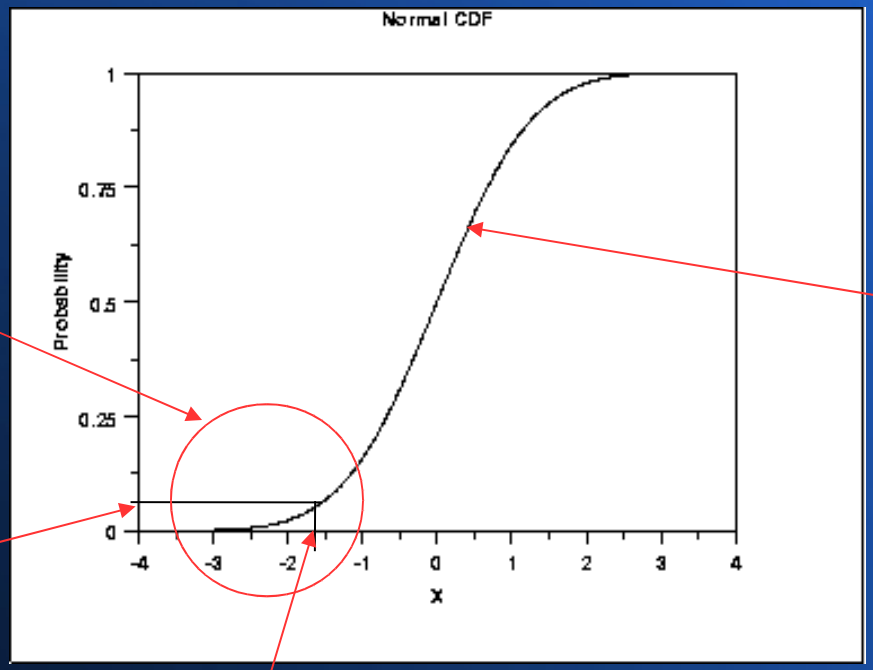
- Financial model developers often face challenges in *integrating new pricing models into risk management systems*
- Financial modelers are domain experts with *limited access to the development and testing environment* for the live risk management system.
- Prototype model migration from a desktop development environment to the deployment environment is often hampered by *disjoint domain expertise*.
- The lag between specification of new models and their deployment results in *lost trading opportunities and unaccountable exposure to risks.*

VaR Methodologies

- **Portfolio based:** *Models the portfolio losses*
  - ✓ computationally more efficient
  - ✘ but assumes a static portfolio loss distribution
- **Asset based:** *Models the asset returns and then **aggregates** the losses*
  - Exact
    - ✓ captures the non-linearity of the portfolio due to market movements
    - ✘ but computationally prohibitive for large portfolios
  - Approximations
    - ✓ remains computationally tractable for large portfolios
    - ✘ but only partially captures the non-linearity of the portfolio

Portfolio Loss Distribution

Tail region represents the most severe losses



Cumulative loss distribution implied from risk returns

1 – confidence level (1-c),

e.g. c=0.95

$$\mathrm{VaR}[-dP] = F^{-1}(1-c)$$

In general, F is not given in analytic form and must be estimated using Monte-Carlo methods