## Altimesh Hybridizer™

Embrace Micro-Architecture Changes Abstract-Out Instruction Set Variety Achieve State-Of-The-Art Performance



## Why HPE ?

- Center of Excellence EMEA located in Grenoble
  - Talented support team
  - Ease of access for pre-GA hardware
- Hardware variety
  - Comprehensive Intel solutions
  - Moonshot platform (ARM)
  - Accelerators AMD and NVIDIA

## Finance and Regulation

- Financial institutions are very creative
  - Derivative products ecosystem grows constantly
  - Some players introduce new product types to leverage corner unregulated financial traits [e.g. Subprimes]
- Every big financial event yields new regulations
  - More stress scenarios [Too big to fail]
  - More complex financial quantitative models [Liquidity]
  - Higher number of simulations [unlikely systemic events]
- Quant analysts need to (re-)design quant libraries constantly
  - New models need to be developed, tested and integrated in existing system
  - Performance is getting critical: from thousands to millions of simulations same power envelope ?
  - Code optimization gets low priority: following changes implied by regulators is already a heavy burden



## Processor Ecosystem

#### • Processors have changed

year	2000	2014	2013	2016	2012	
processor	Pentium 4	Xeon E5-v3	Xeon PHI	KNL	Kepler	
core frequency (GHz)	3,8	2,3	1,24	?	0,745	
vector unit size (DP)	1	4	8	8	32	
pipelines / core	1	2	1	2	2	
contexts	1	2	4	4	4	
core count	1	18	61	72	15	
FMA	1	2	2	2	2	
Peak scalar GFLOPS	3,8	165,6	151,28	375+	22,35	
Peak GFLOPS (DP)	3,8	662,4	1210,24	3000+	1430,4	
SIMD/SIMT ratio	1	4	8	8	64	
Bandwidth (R/W)	4,26	68	352	400+	288	
flop / memop	7,14	77,93	27,51	~60	39,73	
Bandwidth / core	4,26	3,78	5,77	~5,6	19,20	

- Frequency drops, Core count / vector unit explodes
- Most problems get memory bound (flop / memop > 25)
- Multithreading is not the only issue (SIMD/SIMT ratio)
- Keeping-up with technology changes requires significant software development effort and training

## Key Changes to Embrace

- Multithread : core count explode, and frequency stalls or decrease => not using multithread will lead to performance decrease in the future
- Vectorize : vector unit size grows. SIMD/SIMT ratio indicates the relative loss when not vectorizing code. AVX-512 will double the fall for Intel x86 architecture.
- Cache-aware : flop/memop increase (> 25). Operations need to occur in cache. Large vector operations are memory bound and should be replaced by small vector operations

Hybridizer aims at addressing these challenges with a unified approach

## Hybridizer Solution

- Input
  - .Net
  - Java
  - C/C++ (ongoing developments)
- Environments:
  - Windows / Linux
- Generate source code
  - CUDA/C for NVIDIA GPU
  - C++ for native platforms
  - Open CL



## Hybridizer Benefits

#### • Single version of source code

- Express parallelism with a paradigm of choice (ParallelFor / iterators / custom indexing type)
- Generates several flavors of source code

#### • Execution on a variety of platforms

- Plain C, CUDA
- Vector-units: AVX, AVX2, AVX-512
- External libraries integration (e.g. MKL) and extensibility (hand-tuned micro-architecture specific codes)

#### • Debugging / Profiling of output

- Code location is preserved on target platform
- Integration in existing debugging / profiling tools
- Generated source-code is readable for auditing



## Integration with Intel Vtune Amplifier

Name       Name       Name       Approximation       OUT me tori       OUT me tori         International System       Materia       Same       Approximation       Materia       Approximation         International System       Materia       Same       Approximation       Materia       Approximation         International System       Materia       Same       Approximation       Materia       Approximation       A	arger 💥 Analysis Type 🚟 Collection Log 🗖 Summary 💕 Bottom-up 💕 Caller/Callee 😽 Top	o-down Ire	ee 🔤 🔜 Tasks an	d Frames	E BlackSch 💥				
Locs         Address         None         Agendy         OUITen Test	ssembly 📰 🖼 🐼 🚸 🔮 🐑 🔍 Assembly grouping: Address								
Diter states         Disk         Bits 460           Diter states         Disk         Bits 460           Disk         Bits 460         Disk         Disk <thdisk< th="">         Disk         <thdisk< th=""> <thdi< th=""><th>Source</th><th>^</th><th>Address 🔺</th><th>Sour</th><th>Assembly</th><th></th><th>CPU Time: Total</th><th>CPU Time: Self</th><th>*</th></thdi<></thdisk<></thdisk<>	Source	^	Address 🔺	Sour	Assembly		CPU Time: Total	CPU Time: Self	*
<pre>mndaard System.Math methods i</pre>	blic static class BlackScholesMertonPrimitive		0x18000591b		Block 46:				
Indard System. Math methods         Image gene gene gene gene gene gene gene g			0x18000591b	115	vmulpd ymm4, ymm0, ymmword ptr [r13+0xc20]		0.002s	0.002s	
Nuclear out with a state in the state of the state out with a state state out with a state out with a state out with a	ndard System Math methods		0x180005924	115	vmulpd ymm0, ymm7, ymmword ptr [r13+0xba0]	Manna	d <sup>ost</sup> n Intol	CVSIN AI	
intermediate         intermediate<	inuaru System.iviatii methous		0x18000592d	115	vmovupd ymmword ptr [r13+0x500], ymm4	iviappe		<b>DVIVIL</b>	
const double At -1.82253757         Outfoil         Outfoil         Outfoil           const double At -1.82257875         Outfoil         Outfoil         Outfoil           const double At -1.82257875         Outfoil         Outfoil         Outfoil         Outfoil           const double At -1.82257875         Outfoil         Outfoil <td>const double A3 = 1.781477937;</td> <td></td> <td>0x180005936</td> <td>225</td> <td>vmovupd ymmword ptr [r13+0x1c40], ymm4</td> <td>••</td> <td>0.028s</td> <td>0.028s</td> <td></td>	const double A3 = 1.781477937;		0x180005936	225	vmovupd ymmword ptr [r13+0x1c40], ymm4	••	0.028s	0.028s	
cont exclusible M = 1,152/21142/s         Contended by pressure (114-1000)         Note A = 1,10000000000000000000000000000000000	const double A4 = -1.821255978;		0x18000593f	768	<pre>call 0x18000aff8 <svml_exp4></svml_exp4></pre>		0.016s	0.016s	
cdnaf double Storz?         c.008         0.008         0.008           phile File         cdnaf double Stor?         cdnaf double Stor? <t< td=""><td>const double A5 = 1.330274429;</td><td></td><td>0x180005944</td><td>1</td><td>Block 47:</td><td></td><td></td><td></td><td></td></t<>	const double A5 = 1.330274429;		0x180005944	1	Block 47:				
philo state         0.00094         0.001         0.001         0.001           double K = 1.0 / (1.0 + 0.2314419 * Nuch.abe(d))2         0.000946         0.001         0.001         0.001           Much.Exp(-d.5 + d + 0)         0.001         0.001         0.001         0.001         0.001         0.001           (k + 0.1 + k + (0.4 + k + (0.4 + k + (0.4 + k + 0.5))));         0.001         0.0	const double RSQRT2PI = 0.39894228040143267793994605993438;		0x180005944	115	vmulpd ymm4, ymm0, ymmword ptr [r13+0xc60]		0.003s	0.003s	
public disk f = 1.6 / (1.0 + 0.311419 * Math.Mat(d));         00171         00171         00171           ideal = disk f = 350(2727) * (0.0 + 0.311419 * Math.Mat(d));         0000996         23         movedpa ymest, ymel, ymel         00051			0x18000594d	115	vmovupd ymmword ptr [r13+0x4e0], ymm4		0.101s	0.101s	
(       double K = 1.0 / (1.0 + 0.232443 * Math.Mar(d))*       0.008       0.008       0.008       0.008         double cnd = M5g(T2F1 * Math.Mar(-0.5 * 4 + d)       0.008       0.008       0.008       0.008       0.008         f(d > 0)       0.008       0.008       0.008       0.008       0.008       0.008         if (d > 0)       0.008       0.008       0.008       0.008       0.008       0.008         if (d > 0)       0.008       0.008       0.008       0.008       0.008       0.008         if (d > 0)       0.008	public static double CND(double d)		0x180005956	225	vmovupd ymmword ptr [r13+0x1c60], ymm4		0.017s	0.017s	
double k = 1.0 / (1.0 + 0.232443* Math.Abs(d))         0.0051			0x18000595f	763	vandpd ymm5, ymm9, ymmword ptr [r13+0x740]		0.028s	0.028s	
double end = R5gRT2F1 *         0.554         0.554         0.554           if (d > 0)         (f < A)	double K = 1.0 / (1.0 + 0.2316419 * Math.Abs(d));		0x180005968	88	vfmadd213pd ymm5, ymm8, ymm12		0.003s	0.003s	
double end = \$502727 *       0.554       0.554       0.554         Nath_Exc[-0.3 * d + d]       "morphy symmetry [r13*0x170], yms4       0.064       0.066         * (r + (A1 + K + (A2 + K + (A3 + K + (A4 + K + A5)))));       "morphy symmetry [r13*0x170], yms4       0.054       0.056         * (r + (A1 + K + (A2 + K + (A3 + K + (A4 + K + A5))));       "morphy symmetry [r13*0x170], yms4       0.056       0.056         * (r + (A1 + K + (A2 + K + (A3 + K + (A4 + K + A5))));       "morphy symmetry [r13*0x170], yms4       0.056       0.056         * (r + (A1 + K + (A2 + K + (A3 + K + (A4 + K + A5))));       "morphy symmetry [r13*0x170], yms4       0.056       0.056         * (r + (A1 + K + (A2 + K + (A2 + K + (A3 + K + (A4 + K + A5))));       "morphy symmetry [r13*0x170], yms4       0.056       0.056         * (r + (A1 + K + (A2 + K + (A2 + K + (A3 + K + (A4 + K + A5))));       "morphy symmetry [r13*0x170], yms4       0.056       0.056         * (r + (A1 + K + (A2 + K + (A2 + K + (A2 + K + (A3 + K + (A4 + K + A5))));       "morphy symmetry [r13*0x170], yms4       0.056       0.056       0.056         * (r + (A1 + K + (A2 + K +			0x18000596d		vdivpd ymm4, ymm12, ymm5				
Neth, Exp(-0.5 + 4 + 4)       0.056       0.056       0.056         (K + (A1 + K + (A2 + K + (A1 + K + (A4 + K + A5)))));       0.056       0.056       0.056         (K + (A1 + K + (A2 + K + (A1 + K + (A4 + K + A5))));       0.056       0.056       0.056         (f ( > 0)       0       0.057       0.057       0.057       0.057         (f ( > 0)       0       0.058       0.057       0.057       0.058       0.0068	double cnd = RSQRT2PI *		0x180005971	225	vmovupd ymmword ptr [r13+0x1780], ymm4		0.554s	0.554s	
**       *       **       **       **       *       *       *       *       **       *	Math.Exp(-0.5 * d * d)		0x18000597a	763	vandpd ymm4, ymm13, ymm9		0.036s	0.036s	
(K* (A) + K* (A)	*		0x18000597f	_	vfmadd213pd ymm4, ymm8, ymm12				
if (d > 0)       columna       columna <td>(K * (A1 + K * (A2 + K * (A3 + K * (A4 + K * A5)))));</td> <td></td> <td>0x180005984</td> <td></td> <td>vdivpd ymm4, ymm12, ymm4</td> <td></td> <td></td> <td></td> <td></td>	(K * (A1 + K * (A2 + K * (A3 + K * (A4 + K * A5)))));		0x180005984		vdivpd ymm4, ymm12, ymm4				
1 (g > 0)			0x180005988	225	vmovupd ymmword ptr [r13+0x17a0], ymm4	11/V2 ir	offician	0.3395	
end = 1.0 - end;       0.0001       0.0001       0.0001         return end;       0.00000994       0.0000994       0.0001       0.0001         0.00000994       0.0000994       0.0001       0.0001       0.0001         0.00000994       0.0000994       0.0001       0.0001       0.0001       0.0001         0.0000094       0.000094       0.0001       0.0001       0.0001       0.0001         0.0000094       0.000094       0.0001       0.0001       0.0001       0.0001         0.0000094       0.000094       0.0001       0.0001       0.0001       0.0001         0.0000094       0.000094       0.0001       0.0001       0.0001       0.0001       0.0001         0.0000094       0.000094       0.000094       0.0001	if (d > 0)		0x180005991	/63	vandpd ymm4, ymm9, ymmword ptr [r13+0x6e0]		ISTIUCTION	50.020s	
return ond;         (N0000994)         (Window 1942)	cnd = 1.0 - cnd;		0x18000599a	88	ufmadd213pd umm4 umm9 umm12		0.002s	0.002s	
i       University is in the intervent of the inter			0x18000599f	225	valvpa ymm4, ymm12, ymm4		0.507	0.507	
j       Veddpd ymms, ymml,	return cna;		0x1800059a3	225	vmovupa ymmwora ptr [r13+0x17c0], ymm4		0.042-	0.042-	
(double d1 = (indth:Log(pot / strike) + (rate + sigma / 2.0) * maturi         (double d1 = sigmaSqrt7;         (double d1 = (indth:Log(pot / strike) + (rate + sigma / 2.0) * maturi         (double d1 = sigmaSqrt7;         (double d1 = sigmaSqrt7;         (double d1 = sigmaSqrt7;         (double d2 = d1 - sigmaSqrt7;         (double d1 = sigmaSqrt7;         (double d2 = d1 - sigmaSqrt7;			0x180005986	705	vandpd ymm4, ymm15, ymm9		0.001-	0.0435	
Without and the second of t	Manage 1		0x180005961	00	vimaddzispd ynnn4, ynnn5, ynnni2		0.00151	0.00151	
Calar C# source File       00001 might for 00001 might for 00001 might for 00001 might for 011 might f	[Kerner]		0x180005960	225	voivpo ymma, ymmiz, ymma		0.650-	0.650+	
Concerned and an analysis of the strike o	calar (# cource Eile		0x180005958	762	vandad ymme ymmega tr [r13+0x680]		0.049c	0.049c	
double signadgrt1:       double d1 = (shath.Log(spot / strike) + (rate + signa * signa / 2.0) * maturi       double d2 = d1 - signaSqrt1:       double kett = strike + Math.Exp(-rate * maturity):       double kett = strike + Math.Exp(-rate * maturity):       double (NDD0 = CND(d1):       0.037k       0.037k       0.037k         double CNDD2 = CND(d1):       double (NDD0 = CNDD(2):       0       vdivpd ymmk, ymm12, ymm4       0.037k       0.037k       0.037k         gall = spot * CNDD1 - kert * CNDD2:       0       vdivpd ymmk, ymm12, ymm4       0.037k       0.037k       0.037k         gall = spot * CNDD1 - kert * CNDD2:       0       vdivpd ymmk, ymm12, ymm4       0.037k       0.037k       0.037k         gall = spot * CNDD1 - kert * CNDD2:       0       vdivpd ymmk, ymm12, ymm4       0.037k       0.037k       0.037k         gall = spot * CNDD1 - kert * CNDD2:       0       vdivpd ymmk, ymm12, ymm4       0.007k       0.027k       0.027k         gall = spot * CNDD1 - kert * CNDD2:       0       vdivpd ymmk, ymm2, ymm2, ymm4       0.027k       0.027k       0.027k         gall = spot * SIDE = signa * Math.Sqrt(naturity):       0       vdivpd ymmk, ymm2, ymm2, ymm4       0.007k       0.027k       0.027k         gall = signa * Math.Sqrt(naturity):       0       vdivpd ymm4, ymm12, ymm4       0.007k       0.007k       0.007k       0.0001k			0x1000059c5	00	vendpd ynnw, ynnw, ynnword por [113+0x000]		0.00455	0.004	
double d2 = d1 - sigmaSqrt1(nucle + sigmaSqrt2)(nucle + sigmaSqrt2) </td <td>double di = (Marb Log(apor / arriva) + (rara + ajgma * ajgma / 2.0) * maturi</td> <td></td> <td>0x1800059d1</td> <td>00</td> <td>vinaddzi5pd ynnw, ynno, ynniz</td> <td></td> <td>0.0045</td> <td>0.00451</td> <td></td>	double di = (Marb Log(apor / arriva) + (rara + ajgma * ajgma / 2.0) * maturi		0x1800059d1	00	vinaddzi5pd ynnw, ynno, ynniz		0.0045	0.00451	
double kert = strike * Math.Exp(-rate * maturity);double(); <thd>double();double();double()</thd>	double d2 = d1 = signa SartT:		0x1800059d5	225	umound ummord ptr [r13±0x1800] umm4		0.598c	0.598c	
double CHIDD1 = CHID (d1);       double CHIDD1 = CHID (d1);       double CHIDD2 = CHID (d2);       double CHIDD = CHID (d2);       double CHIDD = CHID (d2);       double CHIDD = CHID (d2);       double (d2) = d1, double (f) put, double spot,       double spot,       double chiD (d2);       double (f) = CHID (f);       double chiD (f);       double (f) = CHID (f);       double	double kert = strike * Math Fyn(_rate * maturity):		0x1800059de	763	vandrod ummå ummå umm9		0.037	0.037	
double CMDD2 = CMD(d2);         0.180005968         Vitypd ymm, ymm2, ymm4         0.556         0.556           call = spot * (NDD1 - kert * CNDD2;         0.180005968         225         vmovupd ymmsord ptr [r13+0x620]         0.027s         0.027s           j         0.180005968         Vitypd ymm4, ymm12, ymm4         0.556s         0.566s         0.566s           [Kernel]         0.180005968         Vitypd ymm4, ymm12, ymm4         0.027s         0.027s         0.027s           public static void CallPut(int index, double[] cul, double[] put, double spot,         vitypd ymm4, ymm12, ymm4         0.586s         0.586s           (         0.180005916         Vitypd ymm4, ymm12, ymm4         0.027s         0.027s           double sigmaSqrT1 = sigma * Math.Sqrt(maturity);         0.180005917         225         vmovupd ymmsord ptr [r13+0x180], ymm4         0.001s         0.038s         0.038s           double d1 = (Math.Log(spot / strike) + (rate + sigma * sigma / 2.0) * maturi         0.180005a10         163         vfinyd ymm4, ymm12, ymm4         0.001s	double CNDD1 = CND(d1):		0x1800059e3		vfmadd213pd ymm4, ymm8, ymm12				
call = spot * CNDD1 - kert * CNDD2;       0.1800059ec       225       Vmovupd ymmvord ptr [r13+0x1820], ymm4       0.5565       0.5565         put = kert * (1.0 - CNDD2) - spot * (1.0 - CNDD1);       0.1800059ec       225       Vmovupd ymmvord ptr [r13+0x1820], ymm4       0.027s       0.027s         }       0.1800059ec       0.1800059ec       0.1800059ec       0.1800059ec       0.1800059ec       0.027s       0.027s         [Kernel]       0.1800059ec       0.1800059ec       0.1180005303       0.0112       0.038s       0.038s       0.038s         (kernel]       0.180005307       225       vmovupd ymmvord ptr [r13+0x1840], ymm4       0.586c       0.586c       0.038s         (double sigmaSqrtI = sigma * Math.Sgrt(maturity);       0.180005a10       763       vnahdd ymm4, ymm9, ymmvord ptr [r13+0x1840], ymm4       0.001s       0.001s         0.180005a10       63       vnahdd ymm4, ymm9, ymmvord ptr [r13+0x1840], ymm4       0.001s       0.001s       0.001s         (double sigmaSqrtI = sigma * Math.Sgrt(maturity);       0.180005a16       63       vnahd213pd ymm4, ymm2, ymm4       0.001s       0.001s       0.001s       0.001s         0.180005a10       63       vnahd213pd ymm4, ymm2, ymm4       0.001s       0.001s       0.001s       0.001s       0.001s       0.001s       0.001s <td< td=""><td>double CNDD2 = CND(d2):</td><td></td><td>0x1800059e8</td><td></td><td>vdivrod vmm4, vmm12, vmm4</td><td></td><td></td><td></td><td></td></td<>	double CNDD2 = CND(d2):		0x1800059e8		vdivrod vmm4, vmm12, vmm4				
put = kert * (1.0 - CNDD2) - spot * (1.0 - CNDD1);       0       0.1800059/5       763       vandpd ymm4, ymm9, ymmoord ptr [r13+0x620]       0.027s       0.027s         j       0       1000059/5       763       vandpd ymm4, ymm9, ymmoord ptr [r13+0x620]       0.027s       0.027s         [Kernel]       0       1000059/6       0       vivpd ymm4, ymm12, ymm4       0       0.038s       0.038s         public static void CallPut(int index, double[] put, double spot,       0       0.180005a07       225       vmovupd ymm4, ymm12, ymm4       0.038s       0.038s       0.038s         double sigmaSqrtI = sigma * Math.Sqrt(maturity);       0       0.180005a07       225       vmovupd ymm4, ymm12, ymm4       0.001s       0.001s         0x180005a07       225       vmovupd ymm4, ymm12, ymm4       0.001s       0.001s       0.001s         0x180005a07       225       vmovupd ymm4, ymm12, ymm4       0.001s       0.001s       0.001s         0x180005a22       225       vmovupd ymm4, ymm12, ymm4       0.001s       0.001s       0.001s         0x180005a22       225       vmovupd ymm4 ymm4, ymm12, ymm4       0.001s       0.001s       0.001s         0x180005a22       225       vmovupd ymm4 ymm4, ym12, ymm4       0.001s       0.001s       0.001s	call = spot * CNDD1 - kert * CNDD2:	_	0x1800059ec	225	vmovupd vmmword ptr [r13+0x1820], vmm4		0.556s	0.556s	
b         b         b         b         b         c         vfmadd213pd ymm4, ymm2, ymm4         c         c           (Kernel]         (M10005s03         Vd1vpd ymm4, ymm2, ymm4         0.586s         0.586s         0.586s           public static void CallPut(int index, double[] call, double[] put, double spot,         (M10005s10         762         vmdpd ymm4, ymm2, ymm4         0.038s         0.038s         0.038s           (         (M10005s10         88         vfmadd213pd ymm4, ymm2, ymm2         0.001s	put = kert * (1.0 - CNDD2) - spot * (1.0 - CNDD1):		0x1800059f5	763	vandpd vmm4, vmm9, vmmword ptr [r13+0x620]		0.0275	0.0275	
Image: Note of the system         Im	}		0x1800059fe		vfmadd213pd vmm4, vmm8, vmm12				
[Kernel]         0x880x507         225         vmovupd ymmword ptr [r13+0x1840], ymm4         0.586s         0.586s           public static void CallPut(int index, double[] call, double[] put, double spot, [         o         763         vmovupd ymmvord ptr [r13+0x1840], ymm4         0.088s         0.038s         0.038s           double sigmaSqrtT = sigma * Math.Sqrt(maturity);         0         0.001s         0.001s         0.001s           double d1 = (Math.Log(spot / strike) + (rate + sigma * sigma / 2.0) * maturi         0         vdivod ymmvord ptr [r13+0x1860], ymm4         0.056s         0.056s           0x180005a20         225         vmovupd ymmvord ptr [r13+0x1860], ymm4         0.001s         0.001s           double d1 = (Math.Log(spot / strike) + (rate + sigma * sigma / 2.0) * maturi         0x180005a20         225         vmovupd ymmvord ptr [r13+0x1860], ymm4         0.056s         0.056s           0x180005a20         2         vdivod ymmvord ptr [r13+0x1860], ymm4         0.056s         0.056s         0.056s           0x180005a20         2         vmovupd ymmvord ptr [r13+0x1860], ymm4         0.041s         0.041s           0x180005a20         2         xor r14d, r14d         0.041s         0.041s         0.041s           0x180005a30         vmovupd ymmvord ptr [r13+0xd20], ymm7         vmovupd ymmvord ptr [r13+0xd20], ymm7         vmovupd ymmvor			0x180005a03		vdivpd ymm4, ymm12, ymm4				
public static void CallPut(int index, double[] put, double[] put, double spot,       0       763       vandpd ymm4, ymm9, ymmword ptr [r13+0x5e0]       0.038s       0.038s         (       0       0       0       0       0       0.001s       0.001s         double sigmaSqrtI = sigma * Math.Sqrt(maturity);       0       0       0       0       0       0       0       0.001s       0.001s       0.001s         double d1 = (Math.Log(spot / strike) + (rate + sigma * sigma / 2.0) * maturi       0       0       0       0       0       0       0.001s	[Kernel]	-	0x180005a07	225	vmovupd ymmword ptr [r13+0x1840], ymm4		0.586s	0.586s	
{         0x180005a19         88         vfmadd213pd ymm4, ymm12         0.001s         0.001s           double sigmaSqrtT = sigma * Math.Sqrt(maturity);         0x180005a19         88         vfmadd213pd ymm4, ymm12, ymm4         0.001s         0.001s           double d1 = (Math.Log(spot / strike) + (rate + sigma * sigma / 2.0) * maturi         0x180005a1e         163         vdivpd ymm4, ymm12, ymm4         0.001s         0.001s           double d2 = d1 - sigmaSqrtT;         0x180005a2E         225         vmovupd ymmword ptr [r13+0x1860], ymm4         0.041s         0.041s           double krt = strike * Math.Exp(-rate * maturity);         0x180005a2E         223         xor r14d, r14d         0.041s         0.041s           double CNDD1 = CND(d1);         0x180005a30         vmovupd ymmsord ptr [r13+0xd20], ymm7         vmovupd ymmsord ptr [r13+0xd20], ymm7         vmovupd ymmsord ptr [r13+0xd20], ymm7	public static void CallPut(int index, double[] call, double[] put, double spot,		0x180005a10	763	vandpd ymm4, ymm9, ymmword ptr [r13+0x5e0]		0.038s	0.038s	
double signaSqrtT = sigma * Math.Sqrt(maturity);         br/180005a1e         163         vdivpd ymm4, ymm12, ymm4         0.001s         0.001s           double d1 = (Math.Log(spot / strike) + (rate + sigma * sigma / 2.0) * maturi         br/180005a2e         225         vmovupd ymmoord ptr [r13+0x1860], ymm4         0.566s         0.566s           double d2 = d1 - sigmaSqrtT;         br/180005a2e         223         xor al, al         0.041s         0.041s         0.041s           double kert = strike * Math.Exp(-rate * maturity);         br/180005a2e         223         xor r14d, r14d         0.041s         0.041s           double CDD1 = CND(d1);         br/180005a2e         xor r14d, r14d         br/180005a2e         console (r13+0x420], ymm7         console (r13+0x420], ymm7			0x180005a19	88	vfmadd213pd ymm4, ymm8, ymm12		0.001s	0.001s	
double d1 = (Math.Log(spot / strike) + (rate + sigma * sigma / 2.0) * maturi         0x18005a22         225         vmovupd ymmword ptr [r13+0x1860], ymm4         0.566s         0.566s           double d2 = d1 - sigmaSqrTf;         0x18005a2b         223         xor al, al         0.041s         0.041s         0.041s           double kert = strike * Math.Exp(-rate * maturity);         0x18005a2b         223         xor r14d, r14d         0.41s         0.41s         0.41s           double (DDD1 = cND(d1);         0x18005a2b         230         xor r14d, r14d         0.41s         0.41s         0.41s	<pre>double sigmaSqrtT = sigma * Math.Sqrt(maturity);</pre>		0x180005a1e	163	vdivpd ymm4, ymm12, ymm4		0.001s	0.001s	
double d2 = d1 - sigmaSqrtT;         0x180005a2b         23         xor a1, a1         0.041s         0.041s           double kert = strike * Math.Exp(-rate * maturity);         0x180005a2b         23         xor 14d, r14d         0         0           double CNDD1 = CND(d1);         0x180005a2b         vmovupd ymmword ptr [r13+0xd20], ymm7         0         0         0	double d1 = (Math.Log(spot / strike) + (rate + sigma * sigma / 2.0) * maturi		0x180005a22	225	vmovupd ymmword ptr [r13+0x1860], ymm4		0.566s	0.566s	
double kert = strike * Math.Exp(-rate * maturity);         0x180005a2d         xor r14d, r14d           double CNDD1 = CND(d1);         0x180005a30         vmovupd ymmMord ptr [r13+0xd20], ymm7	double d2 = d1 - sigmaSqrtT;		0x180005a2b	223	xor al, al		0.041s	0.041s	
double CNDD1 = CND (d1); vmovupd ymmword ptr [r13+0xd20], ymm7	<pre>double kert = strike * Math.Exp(-rate * maturity);</pre>		0x180005a2d		xor r14d, r14d				
	double CNDD1 = CND(d1);		0x180005a30		vmovupd ymmword ptr [r13+0xd20], ymm7				
double CNDD2 = CND(d2): 0x180005a39 vmcvdcu vmm/ord ptr (r13+0xd401, vmm10	double CNDD2 = CND(d2);		0x180005a39		vmovdou vmmword ptr [r13+0xd40], vmm10				



## Matrix Multiply

#### Naive Matrix Multiply







## Matrix Multiply

#### Naive Matrix Multiply



# Splitting loops (better cache behavior?)





## Matrix Multiply

#### Naive Matrix Multiply



# Splitting loops (better cache behavior?)



Matrix-Multiply sounds simple, however it involves advanced features:

- Vector-unit operations
- Non-temporal write
- Several layers of memory prefetching
- Many corner cases for unaligned sizes, transposes, etc.

#### **Prefer Vendor-Tuned Libraries**

## A Good Compiler Is Not Enough Use Vendor-Tuned Libraries

- « What every programmer should know about memory », by Ulrich Drepper
  - It takes a lot to write (close to) optimal code
  - Understanding of core components of the system are necessary to get good performance (getting a compute-bound implementation of matrix multiply is hard)
- Micro-architecture evolve
  - AVX means 256 bits operands => new instruction set wrt SSE
  - AVX-2 has more instructions => need to redefine some code (different latencies, fused multiply-add, integer operations, gather instruction)
  - AVX-512 is totally different, moreover flops/memops ratio evolves => need to rewrite
- Vendors provide optimized libraries (Intel MKL)
  - Prefer optimized libraries over hand-written versions
  - Most often better performance writing code to transition from custom data layout to optimized library's data layout
- Hybridizer integrates these libraries with Extensibility attributes
  - Available through wrapper methods (no overhead)
  - No overhead using these libraries
  - Same approach to integrate existing in-house developments

### **ON PERFORMANCE**



## Benchmark-Level Performances



י לוסל	const double A1 = 0.31938153;
	const double A2 = -0.356563782;
438.	const double A3 = 1.781477937;
-50,	const double A4 = -1.821255978;
	const double A5 = 1.330274429;
	<pre>const double RSQRT2PI = 0.39894228040143267793994605993438;</pre>
	double
	K = 1.0 / (1.0 + 0.2316419 * tabs(d));
	double
A5)))));	cnd = PSORT2PT * exp(-0.5 * d * d) *
	(K * (A1 + K * (A2 + K * (A3 + K * (A4 + K * A5))))))
	if $(d > 0)$
	cnd = 1.0 - cnd;
	return cnd;
put,	
st	atic void bsm(double& call, double& put, double 50,
1	double K, double F, double sigma, double T)
L. L	<pre>double sigmaSortT = sigma * ::sort (T) :</pre>
	double d1 = (::log(SO/K) + (r + sigma*sigma/2.0) * T) / sigmaSortT :
	double d2 = d1 - sigmaSqrtT;
	<pre>double kert = K * ::exp (-r * T);</pre>
	<pre>double CNDD1 = cnd(d1);</pre>
	<pre>double CNDD2 = cnd(d2);</pre>
	call = S0 * CNDD1 - kert * CNDD2;
	NDD1) ;
CK-SCHOLES - (	CLOSED FORM
22	
0,402	C++ annotated / Intel Compiler
0,102	DotNet
	Hybridizer
GOPTIONS/S	7% overhead

## Extended features

#### Virtual Functions

- Interfaces / abstract classes and inheritance is supported
- Underlying implementation is a function-table

#### Generics

- Generics get mapped onto templates
- C++ template concepts are expressed by DotNet/Java generics constraints
- Restored performance

Object oriented programming productivity maintained ...

... And overhead can be removed



## Financial Model Spot Diffusion

C++ source code with annotations

(two outer loop configurations)

#### Dot net source code Generic parameters for flexibility

```
[Kernel]
                                                                               oid diffuse (int simCount, int datesCount,
public void Diffusion(
                                                                                           const double* restrict DT,
    int simFrom, int simTo,
    int timeFrom, int timeTo,
                                                                                           const double* restrict brownian,
    Volatility volatility,
                                                                                           double sigma, double rate)
    Rate rate,
    LogSpot logSpot,
                                                                                  #pragma omp parallel for
                                                                                  #pragma simd
    Brownian brownian,
                                                                                  #pragma ivdep
    Schedule schedule)
                                                                                  for (int simId = 0 ; simId < simCount ; ++simId)</pre>
                                                                                     double lnS = logSpot [simId] ;
    for (alignedindex simId = VectorUnit.ID + simFrom;
                                                                                     for (int time = 0 ; time < datesCount ; ++time)</pre>
         simId < simTo; simId += VectorUnit.Count)</pre>
                                                                                         lnS += (sigma * brownian[time * simCount + simId] * sqrtDT[time]) +
                                                                                             (rate - 0.5 * sigma * sigma) * DT[time];
         double lnSk = logSpot[simId, timeFrom];
                                                                                         logSpot[(time+1) * simCount + simId] = lnS ;
         for (int t = timeFrom; t < timeTo; ++t)</pre>
              double sigma = volatility[lnSk, simId, t];
              double sqrtdt = schedule.getSqrtDT(t);
                                                                                   omp parallel fo
              double dt = schedule.getDT(t);
                                                                                int simTo = (th+1) * simCount / 8;
              lnSk += (sigma * brownian[simId, t] * sqrtdt) +
                   (rate[simId, t] - 0.5 * sigma * sigma) * dt;
                                                                                   double* lnS = logSpot + (simCount * time) ;
                                                                                  const double* brow = brownian + (time * simCount);
              logSpot[simId, t + 1] = lnSk;
                                                                                   #pragma ivdep
                                                                                   #pragma simd
```



- Comparing object-oriented code, with generics, processed by Hybridizer
- with hand-written optimized C++ code compiled with Intel Composer 2015



- Hybridizer greatly improves dotnet performance: 5x to 18x
- Object oriented programming preserved: single version of source code, reduces operational risk / testing costs.



- Hybridizer greatly improves dotnet performance: 5x to 18x
- **Object** oriented programming preserved: single version of source code, reduces operational risk / testing costs.
- Hybridizer provides benchmarklevel performances (96% of best performing off-cache)





- Hybridizer greatly improves dotnet performance: 5x to 18x
- Object oriented programming preserved: single version of source code, reduces operational risk / testing costs.
- Hybridizer provides benchmarklevel performances (96% of best performing off-cache)
- Loop ordering has little impact for Hybridizer version (~4%) yet large impact for hand-written implementation (>45%)

NOTE: cache-locality and outer-loop selection has a 10x impact on performance. Writing optimized C++ code requires significant effort and knowledge.

## HOW ABOUT AVX-512 ?



## How about AVX-512 ?

- Hybridizer generates C++ using small vector library (a.k.a. phivect)
- Phivect is implemented and optimized for several microarchitectures
- AVX-512 version of phivect is fully functional.





## Conclusions

- Shortened development cycles
  - Single version of source code with « managed » languages
  - Integrates with Debuggers and Profilers
- State-of-the art performances
  - Software development flexibility without performance costs
  - Close to Benchmark (>90%) for compute and memory bound problems
- Embrace micro-architecture changes
  - Hybridizer is AVX-512 ready simply recompile ?

### http://www.altimesh.com