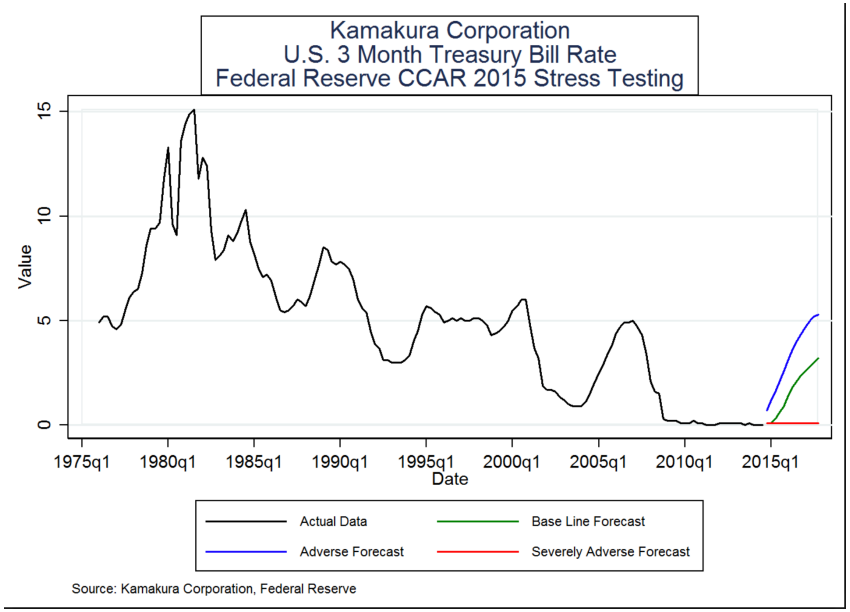
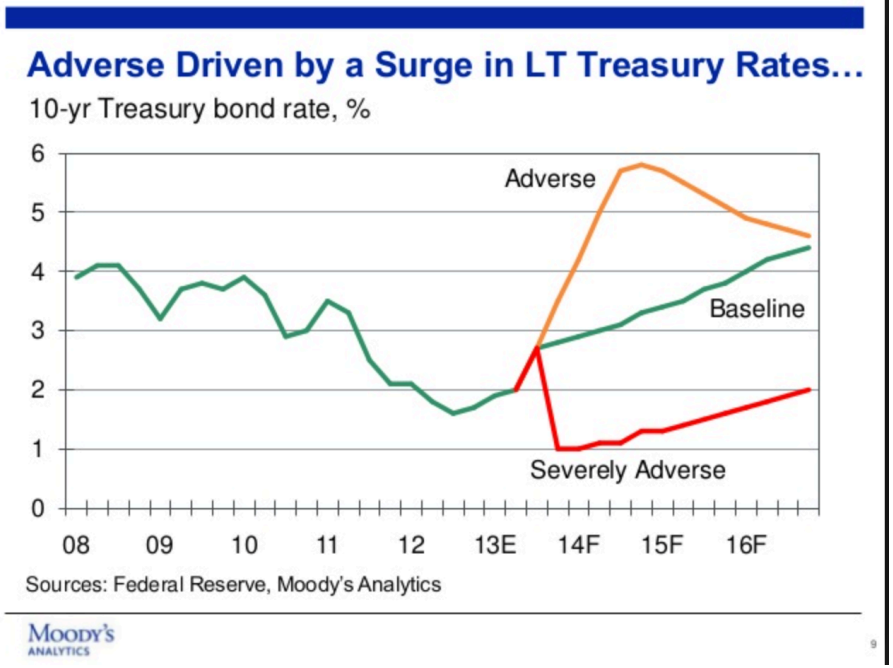

Finding NIMo
Net Interest Margin Optimization
jss
Mar-2015

Main Idea

The Board of Governors of the Federal Reserve System, via the Comprehensive Capital Analysis an Review (CCAR) program, has opened up a new $100+ Bn annual market in optimizing the Net Interest Margins at banks with substantial balance sheets.

CCAR





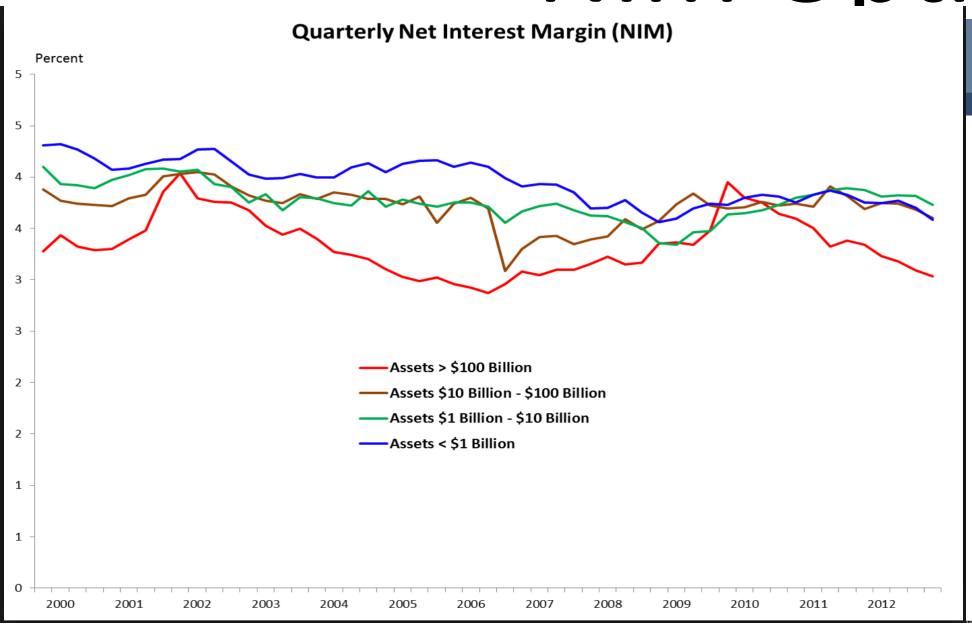
Why is NIM Forecasting important?

- A bp of NIM ~$180mm of annual revenue for large Balance Sheet.
- +0.2% in automated Capital Allocation efficiency at 300 bps NIM is 0.6bps.
- **$100+mm USD annual revenue (in perpetuity), otherwise foregone**.

References:

- Finding Nemo, Disney/Pixar
- van Deventer, Kamakura Corporation, Dec 2014
- FRB, Comprehensive Capital Analysis and Review.
- Assessing the Fed's CCAR Scenarios, Moody's.

NIM Optimization



NIM = Avg. Interest Assets - Avg. Interest Liabilities

Classical Nonlinear Optimization Problem:

Find x in R^n, the allocation of capital to

Maximize:

$$f(x) - \text{The Firm NIM}$$

Subject to:

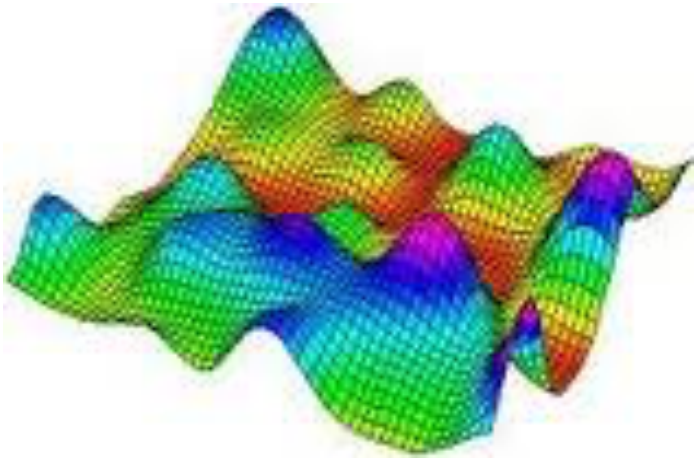
$$g_i(x) <= 0$$

$$h_j(x) = 0$$

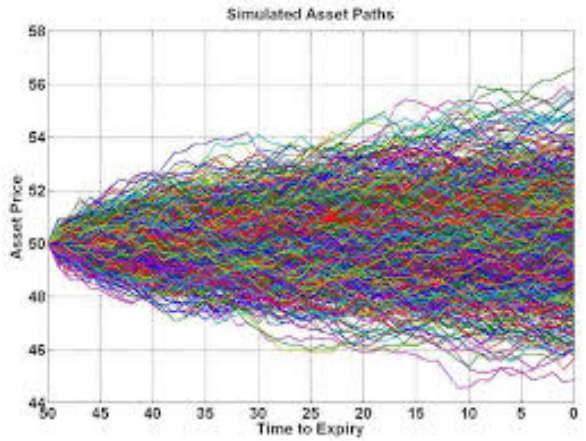
References:

- FDIC, Remarks by Gruenberg 1Q2013
- FRED, NIM for US Banks

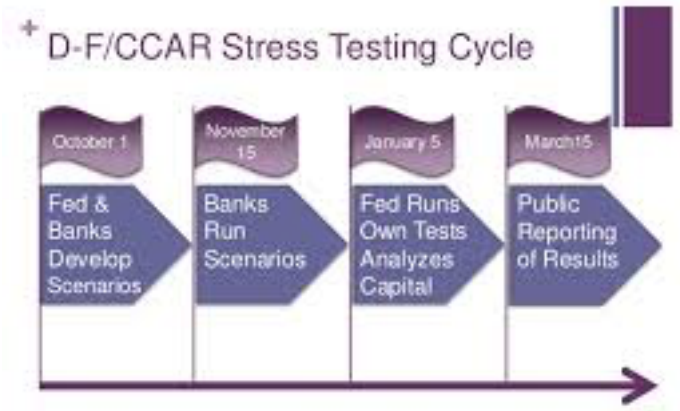
NIMo = NLP over MC over CCAR



NLP – Nonlinear Programming



Monte Carlo



CCAR

References:

http://www.wolfram.com/products/applications/mathoptpro/
http://www.cs.ucsb.edu/~kyleklein/publications/neldermead.pdf
http://www.maths.uq.edu.au/~kroese/montecarlohandbook/
http://www.federalreserve.gov/bankinforeg/ccar.htm
http://www.top500.org/statistics/sublist/

Why is this news?

Why would Top 50 Global Banks be leaving on average $1 Billion of annual revenue on the table? Surely, they must already be Finding NIMo using their own quantitative models.

Industrial Automation

- 80s Trading Pit
- 2000 Bank Floor
- 2010 Colo

Banking >> Amazon

The variation margin is here!

Rate Limiting Factors

1. Different Markets – Different Conventions
2. Multiasset Hedging
3. Trade Contracts imply Term Structures
4. Security Complexity
5. Market Data Complexity
6. Data Quality
7. Multiple Complex Quantitative Models
8. Quant Model Quality
9. Serial Programming

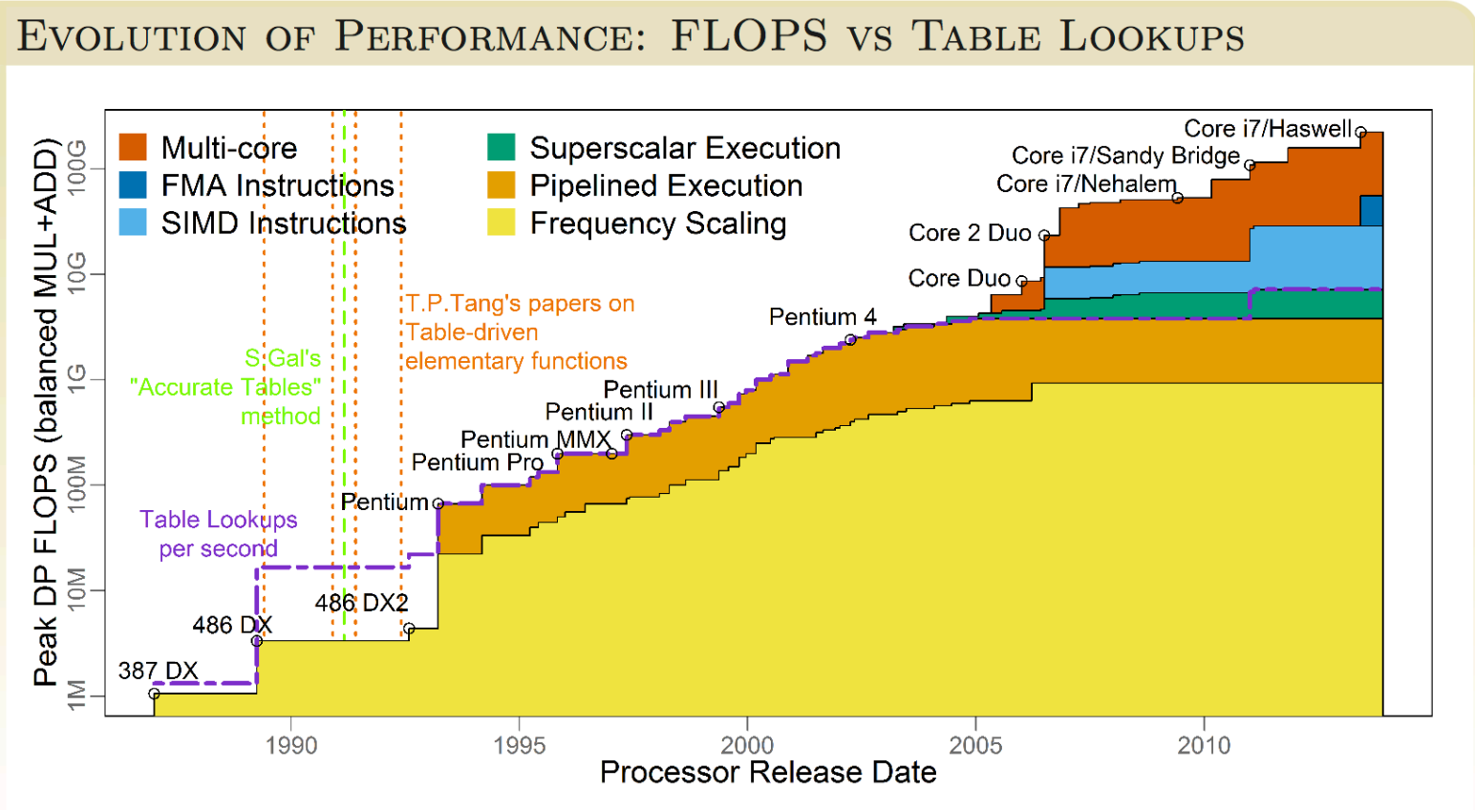
Underneath all this,
10. the Floating Point was slow.

Analogy

Husband and wife commenting on their recent dining experience :

- 'The food was terrible, it was cold, and tasted awful'
- 'and the portions, they were so small'

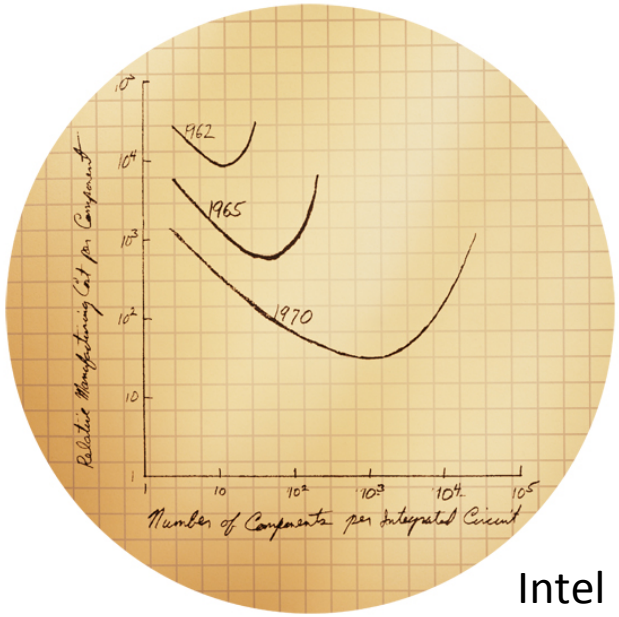
2015 Massive Floating Point Supply



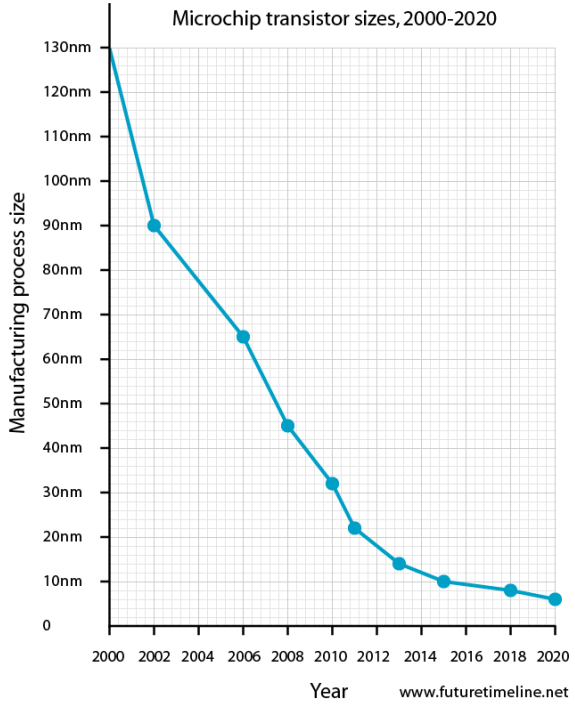
Now just w. commodity microprocessors the 'portions' problem is fixed. The floating point portions are large, very large.

References:
Marat Dukhan, HotChips

Moore's Law



Intel

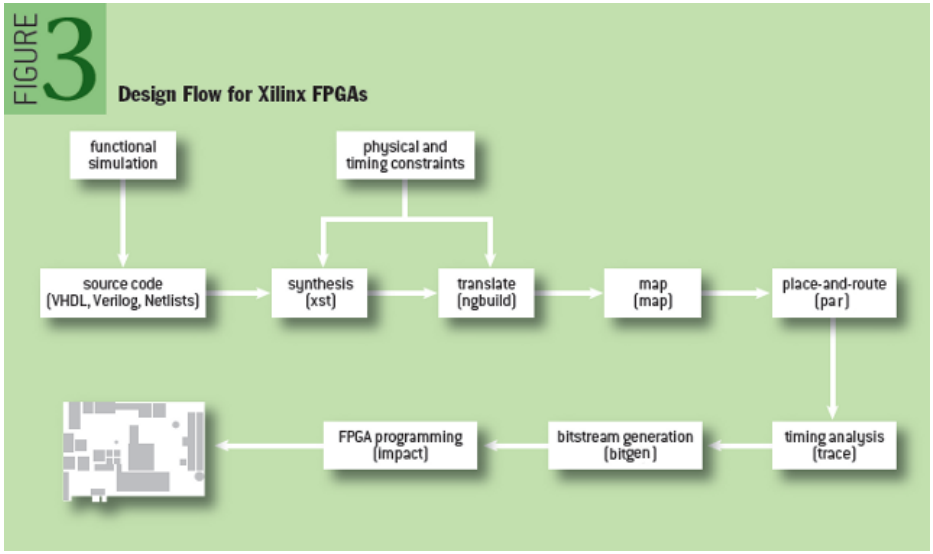


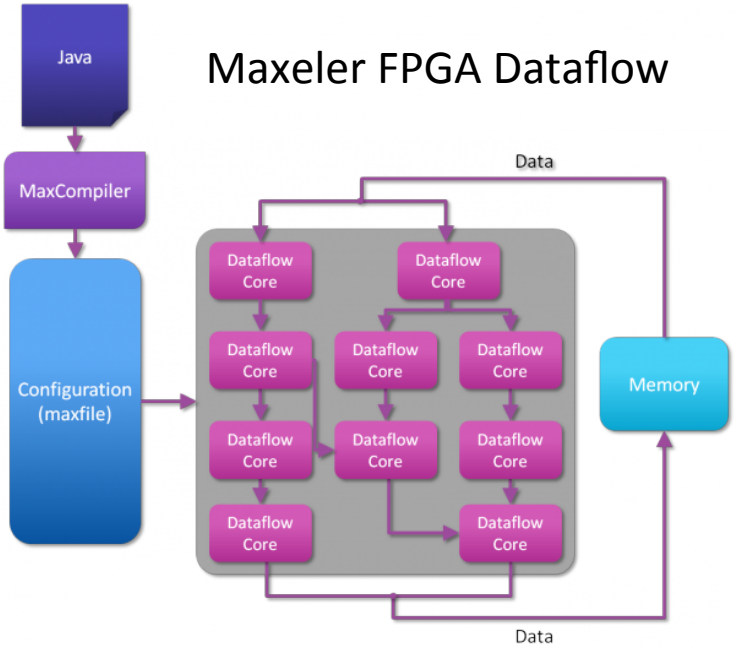
- 3,500x improved clock speed 1980 – pres.
- ~50x improved performance architecture
- FP 'portions' will remain large but perhaps not growing exponentially after 7nm- 5nm ~2020
- Economics could stop Moore's Law before Physics

Moore's Law References:

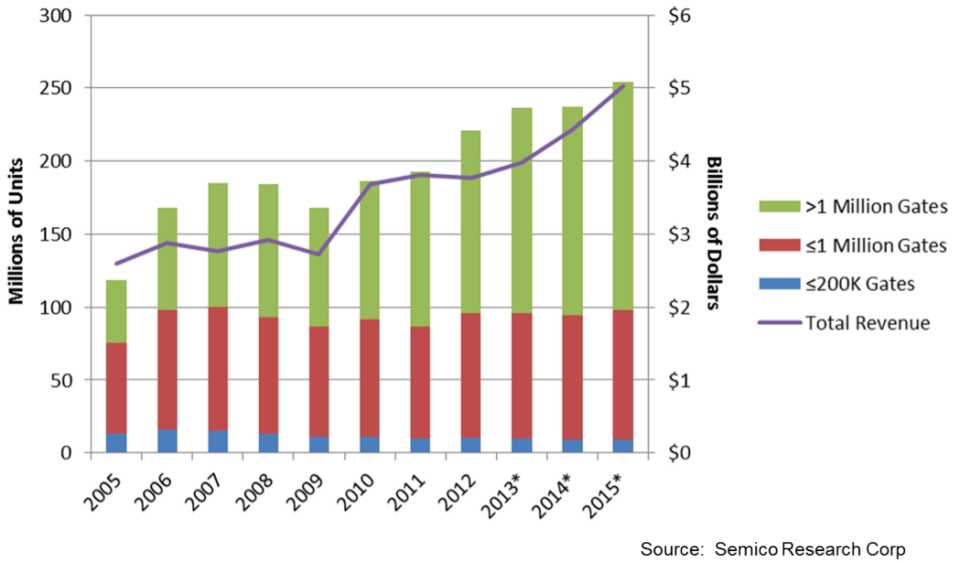
- Colwell, The Chip Design Game at the End of Moore's Law, HotChips

2015 Massive Floating Point Supply





Maxeler FPGA Dataflow

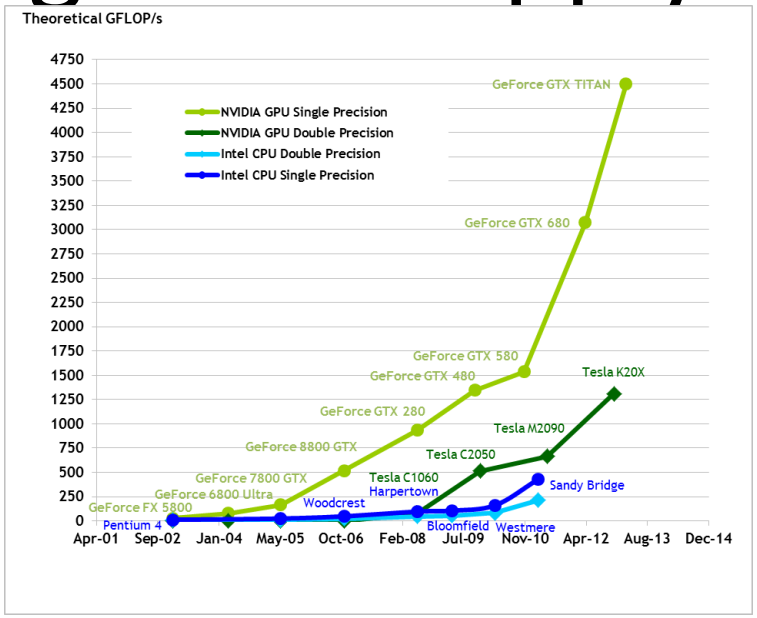


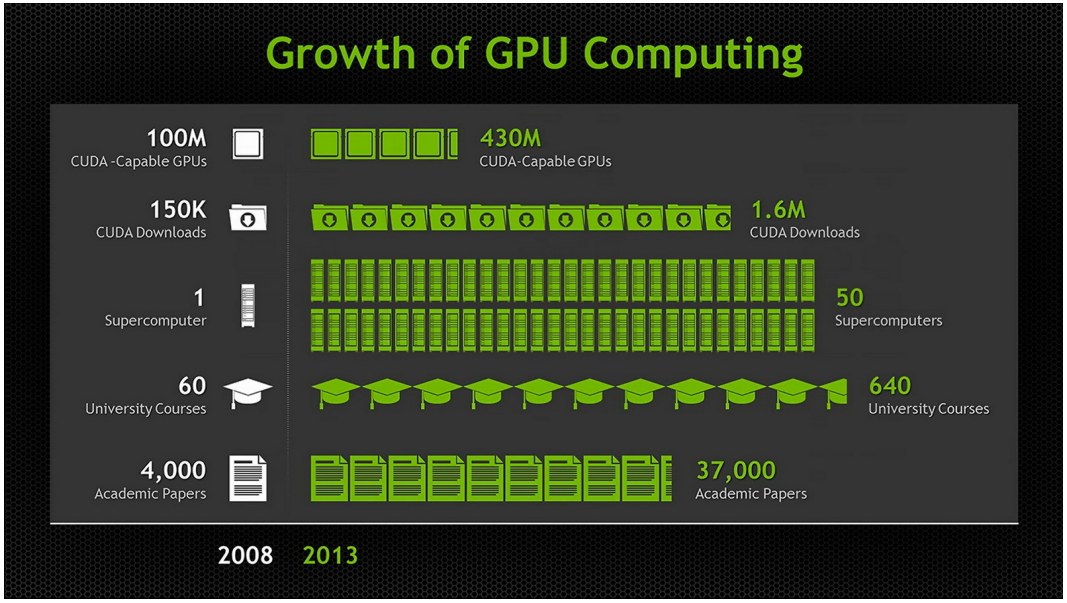
Source: Semico Research Corp

FPGA References:

- Hot Chips2013, Going to the Wire: The next generation financial risk managment platform
- Bacon, et.al., FPGA Programmng for the Masses, ACMqueue, Feb 2013
- Itow, SemiconductorEngineering, Programming The Future, Mar. 2013.
- Tian and Benkrid, Fixed-Point Arithmetic Error Estimation in Monte Carlo Simulations.

2015 Massive Floating Point Supply

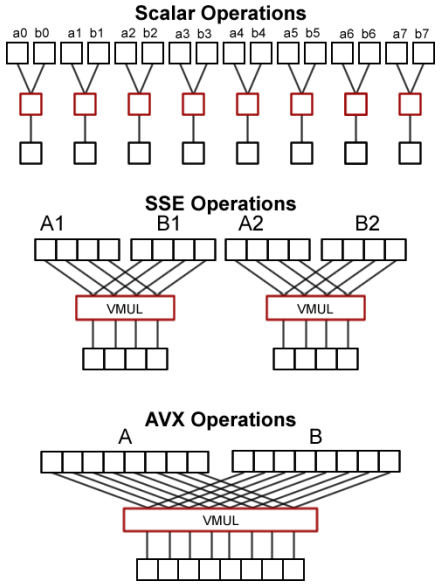


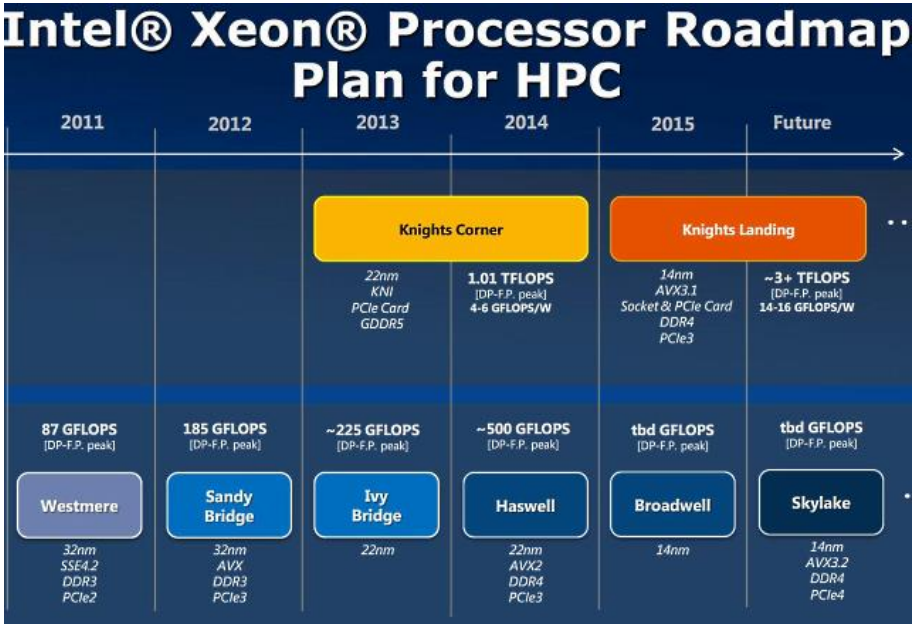


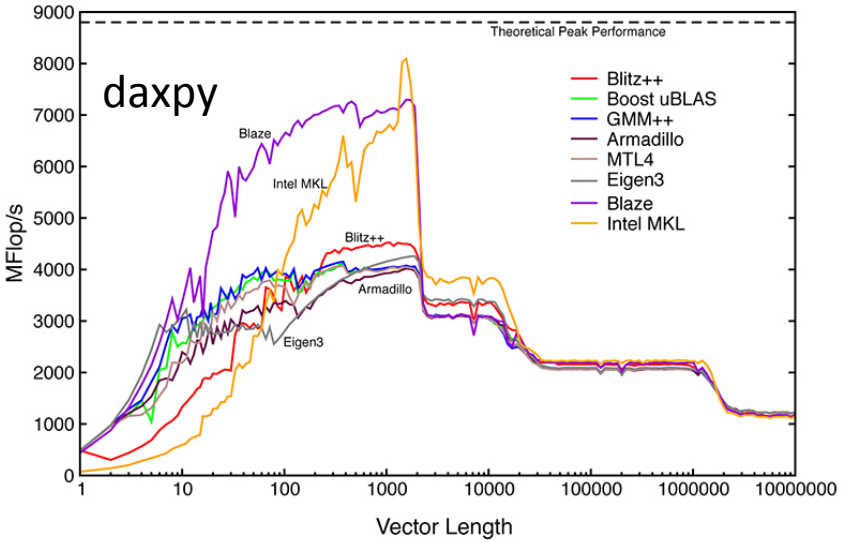
GPU References:

- Gerald Hanweck, Mar. 2013, Accelerating Quantitative Financial Computing with CUDA and GPUs.

2015 Massive Floating Point Supply







x86 References:

- Blaze-lib, code.google.com.
- Intel Roadmap
- Intel Haswell Server chip announcement

Wall Street Applications

Firm Risk

- Evaluate/Aggregate Risk across Firm's entire inventory of Equity, FX, Fixed Income, Commodity securities.
- 10-100 hr. execution window
- 1-10MM positions
- Firm Inventory driven: Mkt Risk, VaR, CVA, FVA
- Monte Carlo

Desk Portfolio P&L

- Evaluate/Aggregate valuation, risk, & explanatories across specific quant models/ trading desks at EOD marks.
- 1-10 hr. execution window
- 100K to 1MM positions
- 10-100x trades
- Security and model driven: Opts, Corps, IRD, CDS, Structs.
- Expression Evaluation, MC

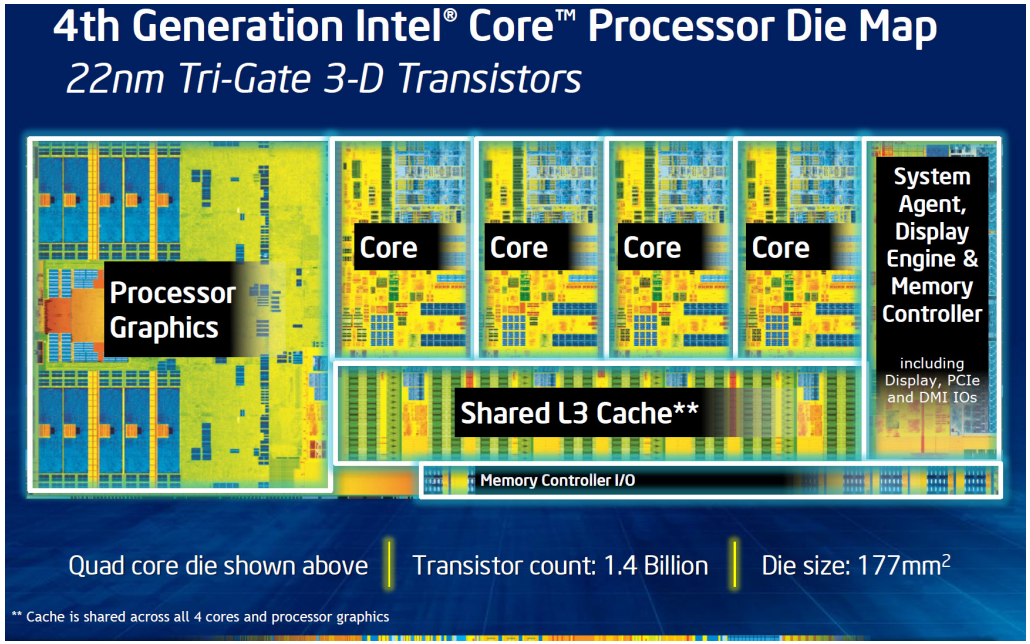
Pretrade Valuation

- Relative value computations, increasingly across multiple asset classes, at intraday marks.
- hrs - 10mics execution window
- Trade/Order driven: Equity, Equity Options, Treasuries, FX, & Futures
- Expression Evaluation

Compliance Risk

- 15c3-5 level per order risk checks, margin opt. across multiple exchanges, SEFs, pools, & ECNs. Trade/Order monitoring & shutdown capacity.
- 1sec – 10 ms execution window
- Max Exposure, Max Dup Orders, Max Capital Exposure
- Expression Evaluation

They Are All Compute Bound



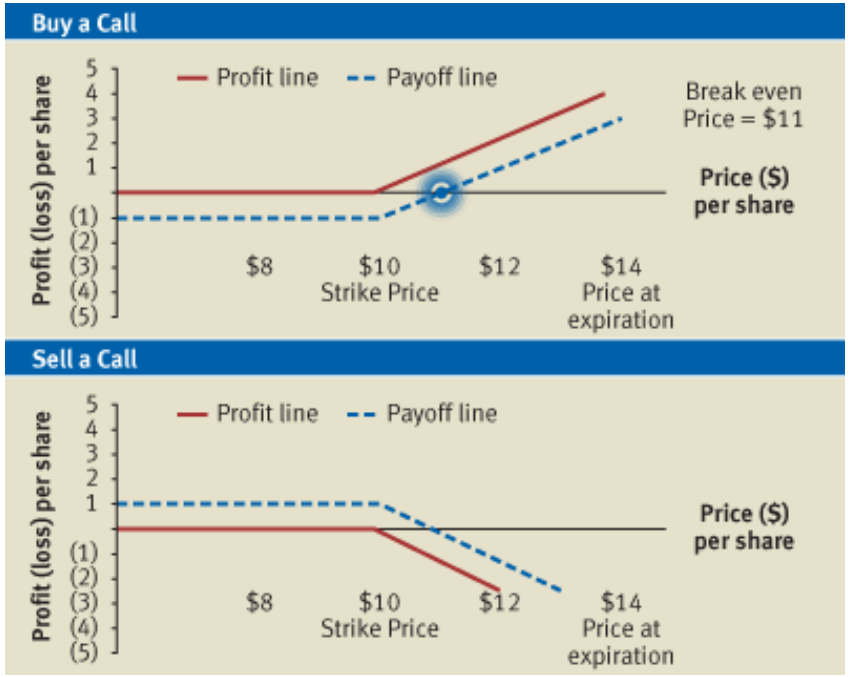
A Compute Bound App may not be competitive:

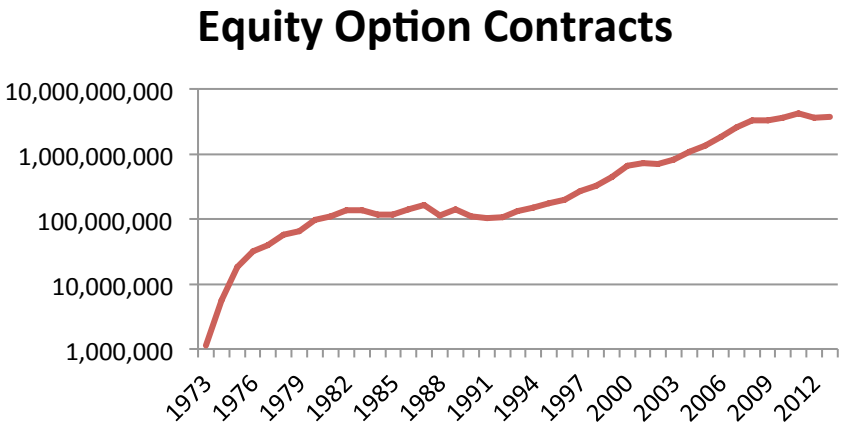
- If the app is ‘ missing’ in L2.
- If the app is running on Haswell but not generating AVX2 vectorized code.
- If the app is compiled –O2 or lower
- If the app was prematurely parallelized.
- If the app uses a serial quant library

Compute Bound References:

- Narayanan, et. al., Generating Performance Bounds from Source Code.

Black Scholes





http://www.theocc.com/webapps/historical-volume-query

$$C(S,t) = N(d_1)S - N(d_2)Ke^{-r(T-t)}$$

$$d_1 = \frac{1}{\sigma\sqrt{T-t}}\left[\ln\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right]$$

$$d_2 = \frac{1}{\sigma\sqrt{T-t}}\left[\ln\left(\frac{S}{K}\right) + \left(r - \frac{\sigma^2}{2}\right)(T-t)\right]$$

$$= d_1 - \sigma\sqrt{T-t}$$

Competitive Black Scholes

- 225MM - 300MM vals/sec, dp on 1 microprocessor in 2014 (~5ns/eval)
- Microprocessor as performance numeraire
- Microprocessor cost $300 – mac book air
- Code, usually the most expensive part, is on the internet, free.
- Compiler – do it the hard way gcc, but it's free. Or the easy way use ICC/MKL, not free.

Black Scholes Application

What is the exact problem to be solved?

- Given n independent double precision inputs to Black Scholes what is the minimum wall clock time required to compute valuation to a known error upper bound of x units in last place (ulp).
- The size of n (number of inputs) and x (size of the tolerable error) constrain the speed of the computation.

Serial Code

```
double callOptionPrice(double S,double t,double X,double r,double sigma,double T)
{
    double d1=(log(S/X) + (r+sigma*sigma/2.)*(T-t))/(sigma*sqrt(T-t));
    double d2=(log(S/X) + (r-sigma*sigma/2.)*(T-t))/(sigma*sqrt(T-t));
    return normalDistribution(d1)*S - normalDistribution(d2)*X*exp(-r*(T-t));
}
```

Test: g++ -m64 –O3 –funroll-loops on an old Westmere MacPro
5890 iteration per ms or 5.89MM evals/sec.

consistent with a different implementation of serial BS by Intel:
“Parallel version runs at 0.00589 Billion option per second:
Completed pricing 1152 million options in 234.880353 seconds:
Figure 1: Code and output of Black-Scholes when compiled with gcc -O2”

Intel Code

```
void BlackScholesFormula( int nopt, tfloat r, tfloat sig,tfloat s0[],
   tfloat x[], tfloat t[], tfloat vcall[], tfloat vput[] ){
vmlSetMode( VML_EP );
DIV(s0, x, Div); LOG(Div, Log); // MKL VML calls
for ( j = 0; j < nopt; j++ ) {
   tr [j] = t[j] * r;
   tss[j] = t[j] * sig_2;
   tss05[j] = tss[j] * HALF;
   mtr[j] = -tr[j];}
EXP(mtr, Exp); INVSQRT(tss, InvSqrt); // MKL VML calls
for ( j = 0; j < nopt; j++ ) {
   w1[j] =(Log[j] + tr[j] + tss05[j]) * InvSqrt[j] *INV_SQRT2;
   w2[j] =(Log[j] + tr[j] - tss05[j]) * InvSqrt[j] *INV_SQRT2;}
ERF(w1, w1); ERF(w2, w2);  // MKL VML calls
for ( j = 0; j < nopt; j++ ) {
   w1[j] = HALF + HALF * w1[j];
   w2[j] = HALF + HALF * w2[j];
   vcall[j] = s0[j] * w1[j] - x[j] * Exp[j] * w2[j];
   vput[j] = vcall[j] - s0[j] + x[j] * Exp[j];}}
```

Cycle Counts: MKL and Agner Fog

Intel ® Math Kernel Library 11.0

List of VML Functions

Real Functions

Trig	Hyper	Pwr Root,	Exp/Lo	Ari	Rounding	Special
Acos	Acosh	Cbrt	Exp	Abs	Ceil	CdfNorm
Asin	Asinh	Hypot	Expm1	Add	Floor	CdfNormInv
Atan	Atanh	Inv	Ln	Div	Modf	Erf
Atan2	Cosh	InvCbrt	Log10	LinearFrac	NearbyInt	Erfc
Cos	Sinh	InvSqrt	Log1p	Mul	Rint	ErfInv
Sin	Tanh	Pow		Sqr	Round	ErfcInv
SinCos		Powx		Sub	Trunc	LGamma
Tan		Pow2o3				TGamma
		Pow3o2				
		Sqrt				

I5-4670	HA Clk	HA ulp	LA Clk	LA ulp
Divide	7.09	0.50	3.02	3.10
ERF	7.25	0.69	6.05	1.33
Exp	5.12	0.51	3.65	1.98
InvSqrt	3.94	0.52	3.70	1.42
Log	6.24	0.50	6.16	0.80

MKL References:

- Intel Math Kernel Library 11.0, Accuracy of VML Functions
- Intel Math Kernel Library 11.0 , Performance of VML (Non-Threaded) Functions
- AIDA64, x86, x64 Instruction Latency, Memory Latency and CPUID dumps
- Agner Fog, Software optimization resources

Methodology

```
void BlackScholesFormula( int nopt, tfloat r, tfloat sig,tfloat s0[],tfloat
   x[], tfloat t[], tfloat vcall[], tfloat vput[] ){ //sum ~45 dp 23 sp 13.5
   ep
DIV(s0, x, Div); LOG(Div, Log);          // 7.09+6.24 dp 1.77+1.81 sp cycles
for ( j = 0; j < nopt; j++ ) {           // 2 cycles 2 FMA units on each clock
   tr [j] = t[j] * r;
   tss[j] = t[j] * sig_2;
   tss05[j] = tss[j] * HALF;
   mtr[j] = -tr[j];}
EXP(mtr, Exp); INVSQRT(tss, InvSqrt);    // 5.12+3.94 dp 1.35+0.97 sp cycles
for ( j = 0; j < nopt; j++ ) {           // 4 cycles
   w1[j] =(Log[j] + tr[j] + tss05[j]) * InvSqrt[j] *INV_SQRT2;
   w2[j] =(Log[j] + tr[j] - tss05[j]) * InvSqrt[j] *INV_SQRT2;}
ERF(w1, w1); ERF(w2, w2);                // 7.25 + 7.25 dp 4.12 + 4.12 sp cycles
for ( j = 0; j < nopt; j++ ) {           // 3.5 cycles
   w1[j] = HALF + HALF * w1[j];
   w2[j] = HALF + HALF * w2[j];
   vcall[j] = s0[j] * w1[j] - x[j] * Exp[j] * w2[j];
   vput[j] = vcall[j] - s0[j] + x[j] * Exp[j];}}
```

Methodology

SPECIFICATIONS-Essentials

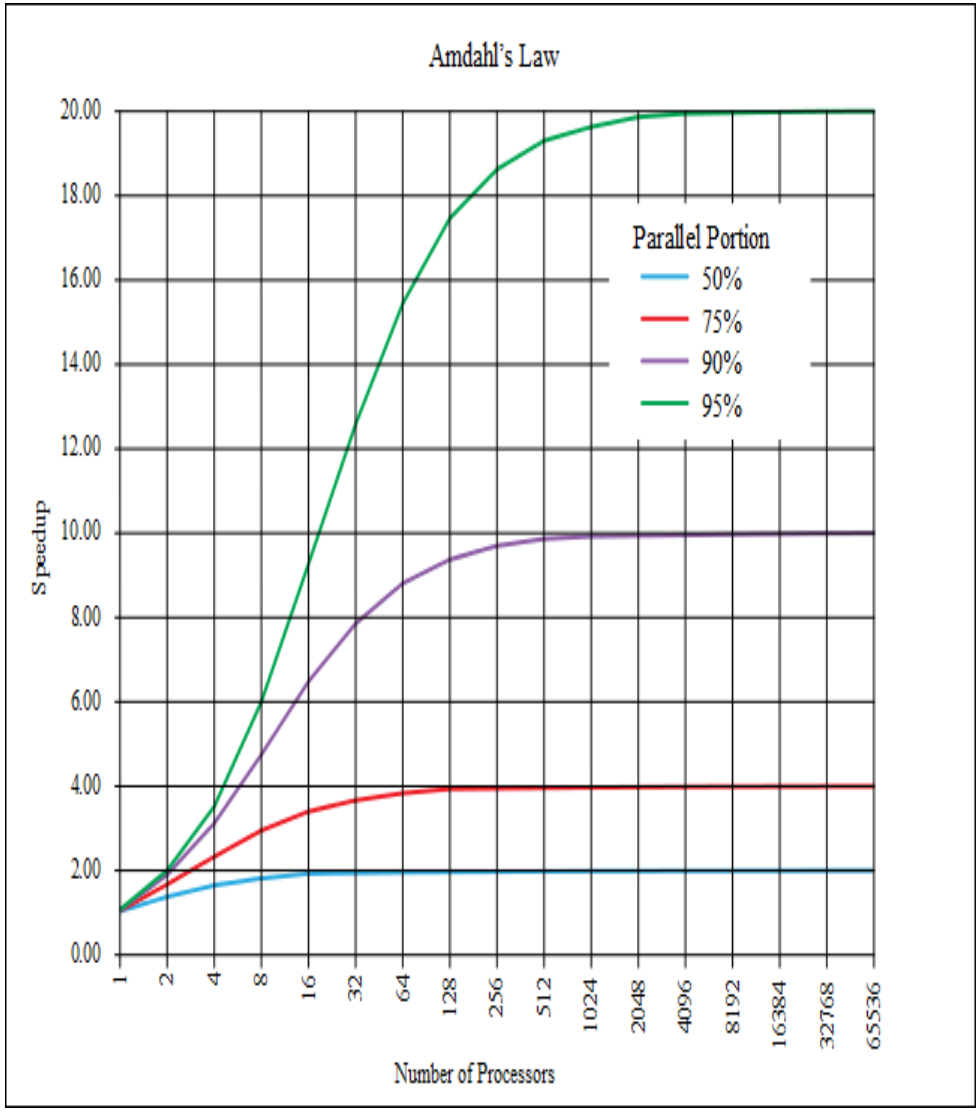
Status	Launched
Launch Date	Q2'13
Processor Number	i5-4670
# of Cores	4
# of Threads	4
Clock Speed	3.4 GHz
Max Turbo Frequency	3.8 GHz
Intel® Smart Cache	6 MB
DMI2	5 GT/s
# of QPI Links	0
Instruction Set	64-bit
Instruction Set Extensions	SSE 4.1/4.2, AVX 2.0
Embedded Options Available	No
Lithography	22 nm
Max TDP	84 W
Thermal Solution Specification	PCG 2013D
Recommended Customer Price	BOX : $224.00 TRAY: $213.00

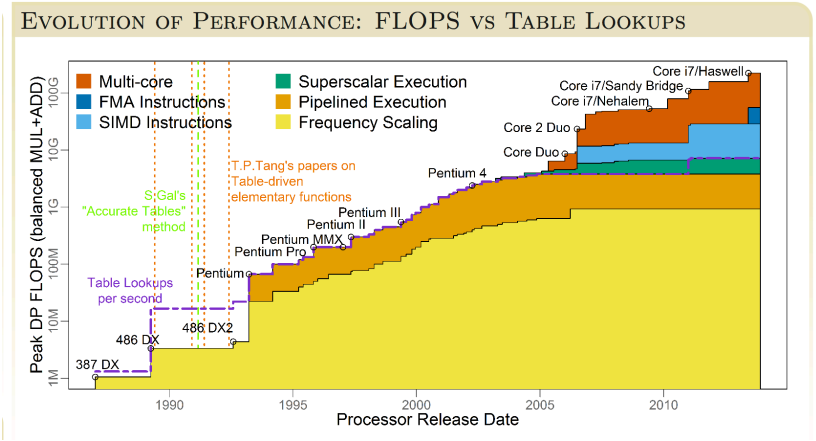
For n = 1000:
dp 45 clocks @ 3.4GHz = 13.2 ns
dp 75MM vals/sec. @1 core
sp 156MM vals/sec. @1 core
ep 250MM vals/sec. @ 1 core
ep est. 250MM vs. Intel 279MM
No threads, No turbo
Assume 90 to 95% parallelizable
Amdahl's Law 3*75MM vals/sec.
225MM to 750MM vals/sec @1 chip
Maxeler ~1B vals/sec. @1 DFE node
AVX2 perf. 12.7x vs. –O2 for dp

Methodolgy References:

- Intel Core i5-4670 Processor – spec sheet.
- Shuo-Li, Intel, Case Study: Computing Black Scholes with Intel Advanced Vector Extensions.
- Zahradnicky and Lorencz, Acta Polytechnica, 2010, FPU-Supported Running Error Analysis.
- Higham, Accuracy and Stability of Numerical Algorithms.
- Muller, 2009, Handbook of Floating-Point Arithmetic.
- Maxeler, Stanford EE380 Jan 2013.

Competitive Performance





Marat Dukhan, HotChips

Competitive Performance References:

- Hager, Fooling the masses with performance results on parallel computers
- Marc Snir, Supercomputing: Technical Evolution & Programming Models.
- Gene Amdahl, AFIPS, 1967, Validity of the single processor approach to achieving large scale computing abilities.

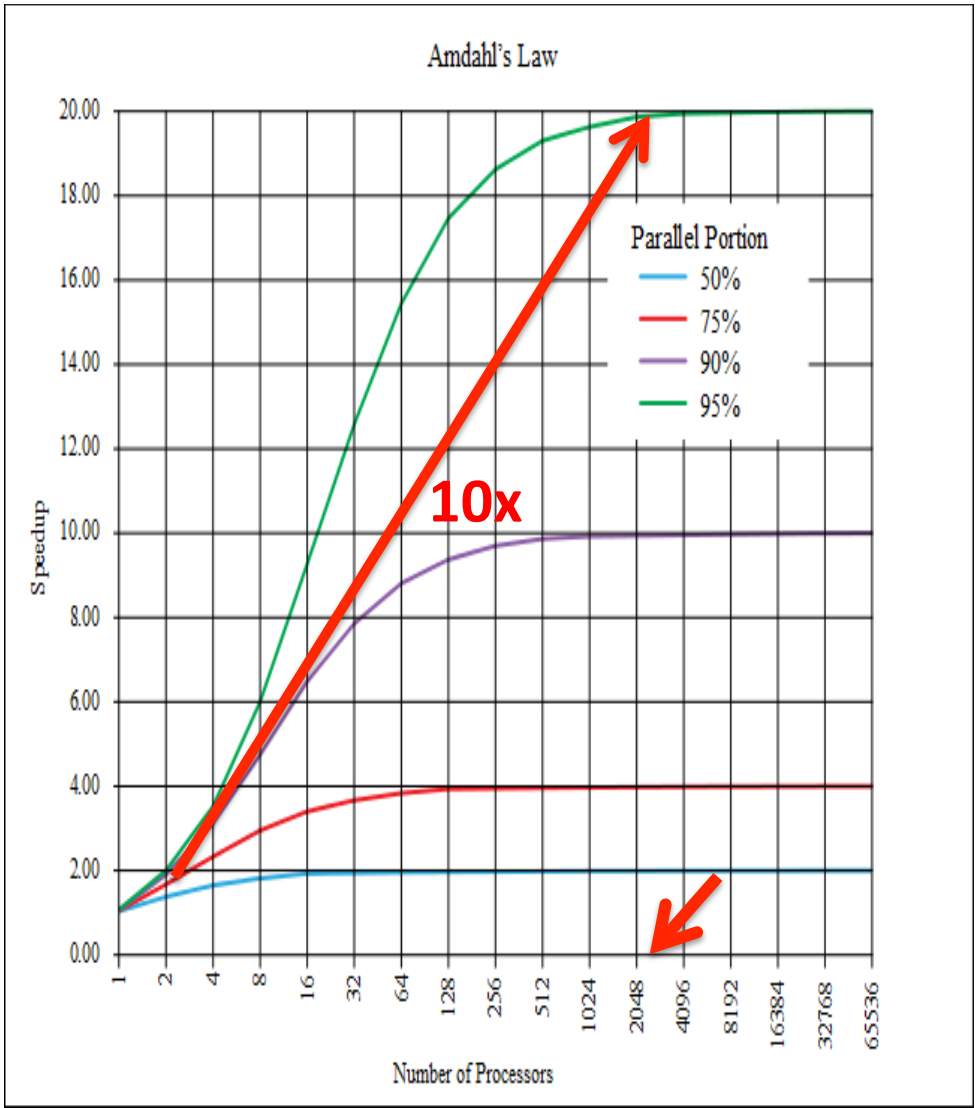
When is Competitive Performance Important?

- When the simulation size is massive.
  - Firm Risk: Monte Carlo
  - Finding NIMo
- When a fixed amount of computation needs to be executed in a small window of time.
  - Pretrade valuation
  - Compliance Risk
  - P&L to a lesser extent

Monte Carlo Example

- Suppose you are running Firm Risk MC on a grid with a vendor supplied serial quant library implementing Black Scholes or Garman-Kohlhagen on a large inventory.
- Assume, for example, 95% of your MC code execution was in the parallel portion (refer to Amdahl's Law graph).

Monte Carlo Example



- The serial quant library is costing you the annual charge back for every grid node after the second one. Those 12.7x speed up cycles are non recoverable (at 95% parallel portion), your MC can only speed up an additional 10x after the second core by allocating more grid nodes.
- All cores running BS in production at < 10% capacity.
- Cost to you (2048-2) core charge back ~ $2MM/yr.
- Split the cost with your serial quant lib vendor?

NIMo Details

So, how would you go about Finding NIMo?

NIMo NLP

Approach: CCAR worst case analysis is extended to Monte Carlo expected case full balance sheet simulation (on a randomly perturbed CCAR base case scenario). The Bank's CCAR infrastructure provides clean Accrual Portfolio data and a Bank/Regulatory framework for reviewing the Balance Sheet simulations. The $g_i(x)$ in the NLP are from the Monte Carlo of the full balance sheet simulation. The NLP checks the outputs from MC for the various incremental capital allocation plans in X (below) and guides the gradient climbing search to the risk adjusted optimal NIM (consistent with the market expectations).

- X contains O(10K) elements
  - NIM output
  - Firm New Investment levels (O(1)) inputs
  - Firm & Regional Risk level constraints (O(100)) outputs
    - Libor, Sovereign, FX, Credit, Vol, and Basis
  - Regulatory level constraints (O (10)) outputs
    - Liquidity Coverage Ratio
    - Net Stable-Funding Ratio
  - Business Entity Balance level constraints (O(10K)) inputs
    - Acquisition, Retention, Runoff goals per GOC (Business Unit)
    - Current Balance Sheet broken down by Business Unit is an input
  - New Product Model Constraints (O(1000)) inputs
    - New origination
- Constraints are sparse
- Parallel Simplex formulations may scale up see: http://www.cs.ucsb.edu/~kyleklein/publications/neldermead.pdf

FLOPS Estimates

New Supercomputer Category like: Weather, Oil, Fluid Flow, Energy, Big Data. 100 TFLOPS @$700K runs for one week to find NIMo. Cray XC @100 PFLOPS finds NIMo in 10 minutes.

- 64 Giga FLOPS per Balance Sheet NIM valuation
- 10K Paths for MC with Risk sensitivities
  - 640 Tera FLOPS
- 100K to 1M NLP vals. to find NIMo, risk weighted
  - Between 64 and 640 Exa FLOPS (if it scales)

Pros and Cons

Pros:

- **High ROI**: ~1 Bn USD per Top 50 Bank Annual Revenue (+3%@300bps) for modest infrastructure and hardware code.
- **Cred w FED**: Capital planning of Firms without competitive Balance Sheet simulation code could reasonably be viewed by The FED as 10 years behind.
- **US Bank NIM at 30 Year lows**: according to FRED. A bp of US Bank NIM value at premium in 2015.
- **NIM_{opt} rides Tech Wave**: Moore's Law continues to improve NIM_{opt} performance (capacity to search) through 2020.
- **Leverage CCAR**
  - cleans the Accrual inventory dataflow, pricing, and indicatives for NIM_{opt} to reuse.
  - CCAR Balance Sheet gives frame of reference for NIM simulation to Treasury users globally.
  - Turns CCAR sunk costs into a Revenue Center.
  - Moves from CCAR worst case analysis to expected case analysis.

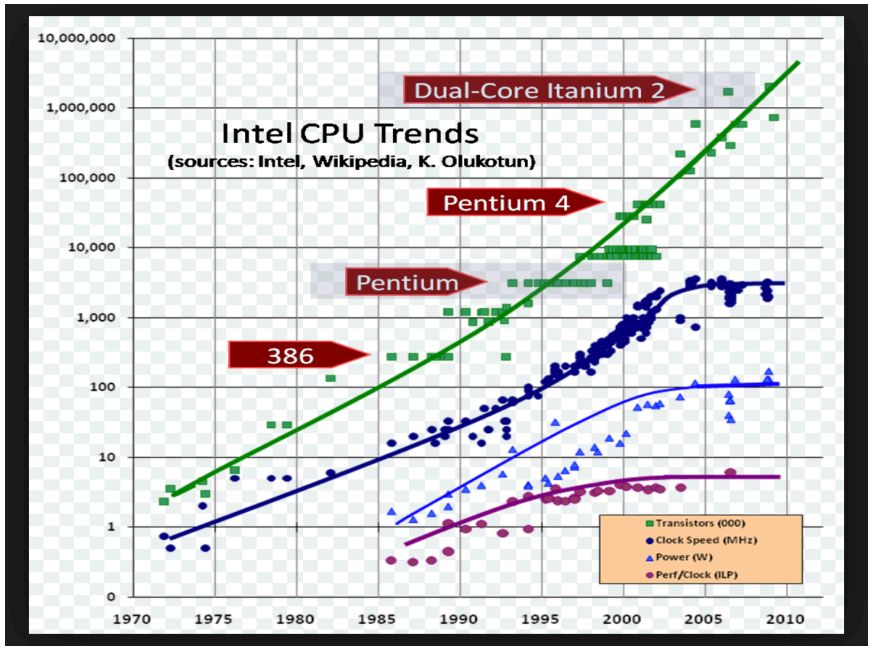
Cons:

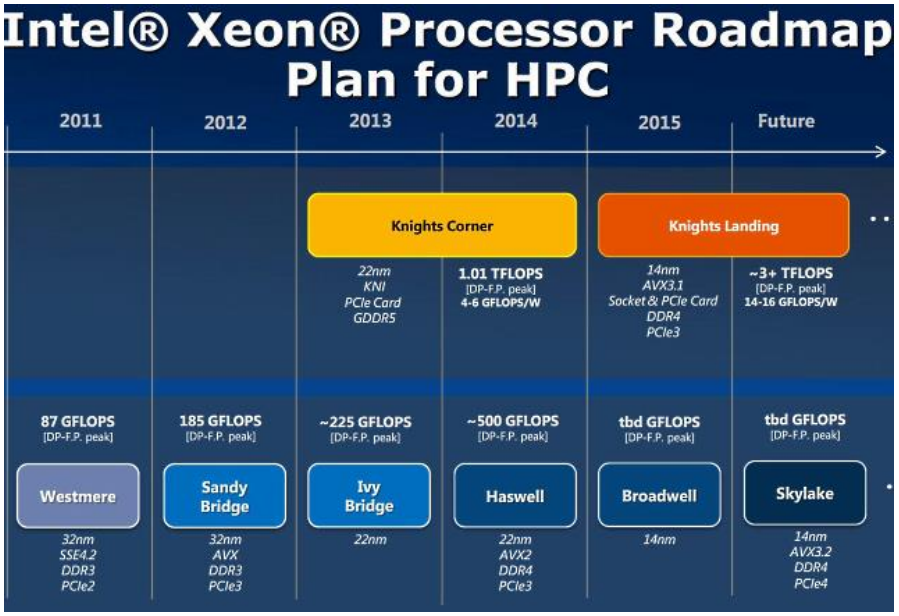
- Could significantly raise the bar on Capital Planning Review at the Federal Reserve once they figure out NIMo technology.
- Need the 1000x kick in efficiency from 2015 FP code optimization to get NIM_{opt} feasible.
- Floating Point optimization literate programmers on the Street are rare.
- NLP/MC is not a trivial math programming problem.
- Maybe Transfer Pricing Curves take some or all of the NIM_{opt} improvements off the table.
- Some parts of the Balance Sheet AFS/HTM costly to simulate without rewriting code not in the Bank's possession.
- Deposit and Loan Modeling needs reasonable fidelity for meaningful/actionable NIMo convergence. If fidelity is too bad, are the CCAR test results really meaningful?

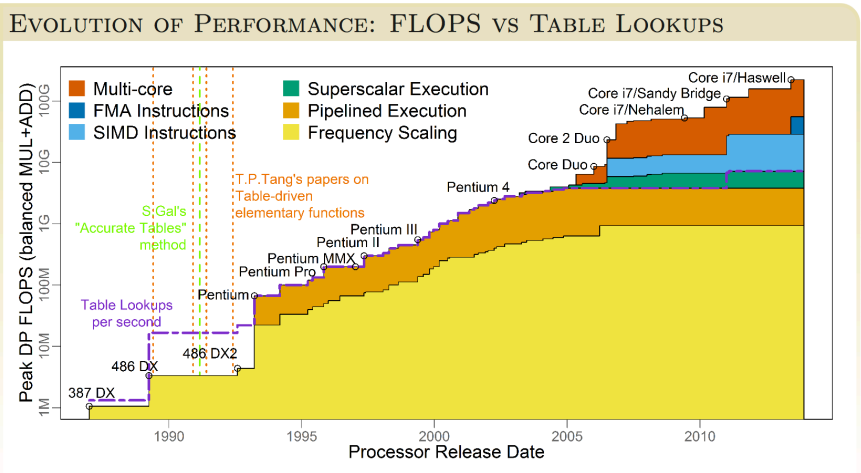
References:

- Sandberg, Citigroup, Ruby Floating Point 2015.
- http://www.computerworld.com/article/2847865/us-sets-sights-on-300-petaflop-supercomputer.html

2015 Inflection Point – "Free" FP



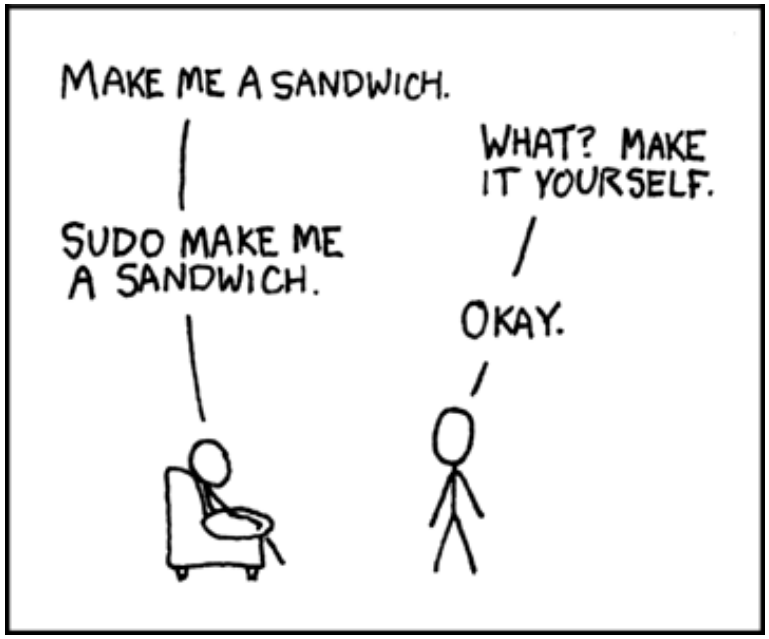




References:

- Dukhan, Hot Chips 2013.
- Intel Roadmap
- Colwell, Hot Chips, 2013, The Chip Design Game at the End of Moore's Law.

Serve Them While They're Hot



http://imgs.xkcd.com/comics/sandwich.png