The Need For Speed

Using Exotic Hardware For Application Performance

John S. Nolan (stigmergist@gmail.com)

My (Technology) Passion

"Turning the cutting edge into the bleeding edge"

Objects

Distributed Computing

Self-Organizing Systems

Subjectivity

AI

Agile

Smart Dust

Synchronous Programming

OBJECTIVE

- **My Objective:**
  - To provide information to “software guys” about the currently available, practical options for performance enhancement through the application of “exotic” hardware
- **Your Objective:**
  - Understand the subject more in order to inform strategy with respect to research and development projects
  - Find out what the practical options are with “cool hardware”
  - Spend sometime in a sitting down whilst waiting for a more interesting session...

Who Am I? And why I'm I Qualified To Be Here?

- **John S. Nolan**
  - "Applying the cutting-edge to make it the bleeding-edge"
    - Have been on the fore-front of introducing objects, distributed systems, agile, smart-dust, exotic hardware...
    - Delivery, Delivery, Delivery...
  - 10+ years working in Financial Industry
    - Employed at JP Morgan, Morgan Stanley, Dresdner Bank
    - Consulted with ABN AMRO, Merrill Lynch, MAN, Standard Bank, UBS, Tullett Prebon, Barclays, etc.
    - Broad experience across fixed income, equities, FX; front-, middle- and (some) back-office
  - 10+ years working in other industries, including
    - Principal Researcher for Sun on low power, high performance platforms (battery powered with a hybrid software/hardware approach)
    - Engineer at BP working on combined hardware/software solutions for fault tolerant control systems for safety-critical processes
  - ACM Distinguished Engineer and Royal Society Innovation prize winner
  - Have done research and practical work with exotic hardware in financial services including equities, CDOs, matching optimization, low-latency systems, etc.

What Do We Mean By "Exotic" Hardware?

- **Technologies outside the experience of most software developers**
  - Not general purpose hardware
    - E.g. Standard server and desktop systems – CPU, memory, storage, IP network, OS
  - Usually require specialist programming skills/tools
  - Usually exist as peripherals in general purpose machines or specialized stand-alone machines
- **Are not easily available or accessible**
- **Are not "user" or "developer" friendly**
- **Are not necessarily aimed at software or application development**
- **Are not easily integrated into a technology strategy**

WHY SHOULD I CARE? WON'T CPUS SAVE THE DAY?

- **CPUs are increasing performance steadily but with some hidden secrets**
  - Processor speeds have increased by 40x over last 10 years but heat density has increased by 60x. Manufacturers are struggling with physical limitations of gate size and density vs. heat output
  - Multi-core processors of stated GHz speeds actually run the separate cores at lower rates to maintain a heat-profile. Consequently, single-threads will actually take longer to execute on multi-core than equivalent single-core (though processor throughput is greater)
  - The "memory wall" – memory access times are being exceeded by processor speeds causing CPUs to be essentially idle, and so inefficient
  - OS and application software models are inefficient at using the hardware effectively, causing the CPU to spend unnecessary time processing OS/management software rather than processing data
    - See recent alterations in scheduling schemes such as Apple's GCD
- **CPU manufacturers are slowly incorporating "exotic" hardware into processors to alleviate this issues**
  - ...but what programming model? How will it impact software?

What "exotic" hardware Can Give you NOW

- **"More Bang For Your Buck"**
  - better performance/price point
- **Dedicated processing**
  - No OS processes or extraneous tasks reducing efficiency
  - Dedicated co-processing – keeping the CPU free for other tasks
- **Lower Heat Output & Power Usage**
  - (generally, but not always true)
  - Greater processing power for the same power/heat profile
  - Approx. 70% of data centre costs are power and cooling requirements – so exotic hardware allows you to increase your processing capacity without increasing your data centre running costs (theoretically!)
- **An upgrade path for existing computers to act as more powerful compute nodes**
- **A jump start on technology that will become general purpose within 3-5 years**

What kind of problems are good targets?

- **Volume problems**
  - Brute force approaches to solve analysis problems
    - The number of Monte Carlo simulations possible in a finite time
    - The number of combinations time of trades/positions testable in a finite to produce an optimal hedge/outcome
    - Testing algorithmic trading algorithms across large numbers of market scenarios
  - Volume of data problems
    - Re-valuing all positions over-night
    - Intra-day enterprise risk
- **Latency problems**
  - Time to react to real-time data events
    - High-frequency trading
    - Temporal arbitrage
    - Cross-market trading
    - Continuous stream processing (i.e. Not batch, not frequent re-run)

What is a good solution ?

- **Fast**
  - How long does it take to get a result from the system?
    - Metric: Measure time and volume figures
- **Soon**
  - How long before you get a working system from the developer?
    - Metric: Measure how long it takes to create/change a solution
- **Affordable and Sustainable**
  - "Bang For Buck" – Capital investment vs. Performance benefits
    - Metric: should be less than $15k a seat (dev h/w + s/w) (for example)
    - Metric: should produce at least a 8x enhancement (for example)
  - Running Costs – Heat, Power, Personnel
    - Metric: heat/power calculations +/- by processing power
    - Metric: cost of staff and/or training

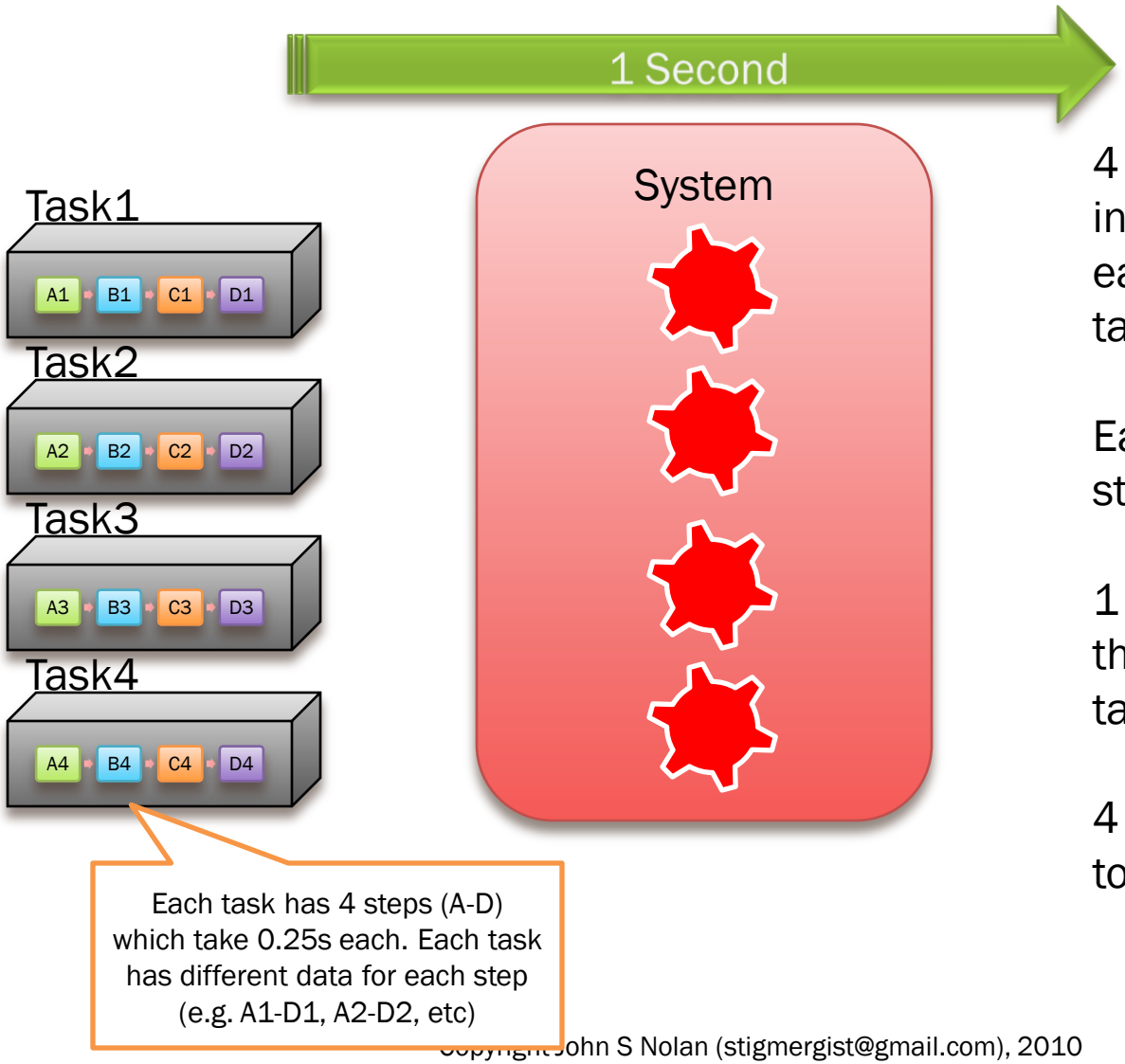
What are the Current Practical Options?

- **Array Processors**
  - SIMD processors with 100s of parallel elements doing the same processing of different data (e.g. Clearspeed)
- **Using GPUs as Array Processors**
  - Graphics Processing Units (GPUs a.k.a. graphics cards) are specialised array processors for graphics – some can be re-purposed for general purpose calculation (e.g. Nvidia)
- **FPGAs**
  - Field Programmable Gate Arrays (FPGAs) (e.g. Xilinx)
  - Dynamically re-configurable chips – the connections between the logic gates within the chip can be dynamically altered
- **Dedicated compute clusters**
  - High-speed interconnect – Grid/Distributed Processing
- **Special Purpose Processors**
  - E.g. NFPs (Network Flow Processors)

Understanding the difference

- **Each of these hardware solutions uses a different performance 'strategy'**
  - Multi-core CPU = Task Parallelism
  - Array Processor = Data Parallelism
  - FPGA = Pipelining
- **Characterising Performance**
  - Absolute Time
    - Time to complete a single task
    - e.g. It takes 1s to calculate the price of a European option
  - Throughput
    - Number of tasks completed by a system per unit time
    - e.g. It can price 4 European options a second
  - These example statements can both be true

Multi-Core CPU: Task-Parallelism



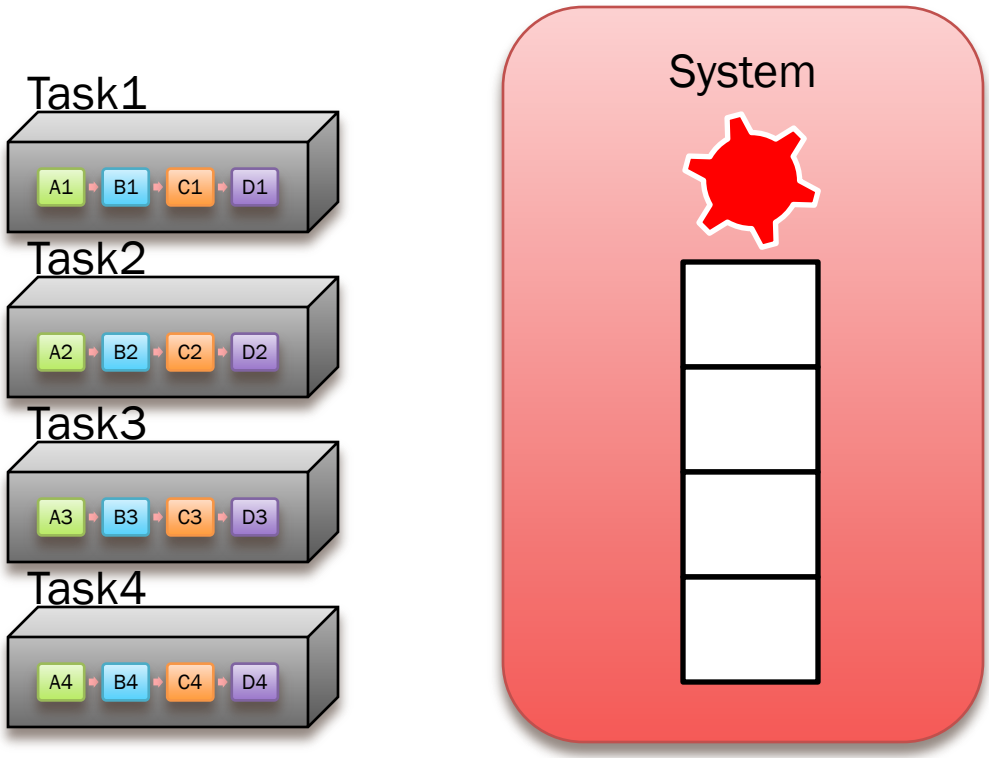
4 processing units inside the system can each undertake any task.

Each processor does all steps A,B,C,D

1 task takes 1 s, so throughput is 4 tasks/second

4 results appear together every second

Array Processor: Data-Parallelism



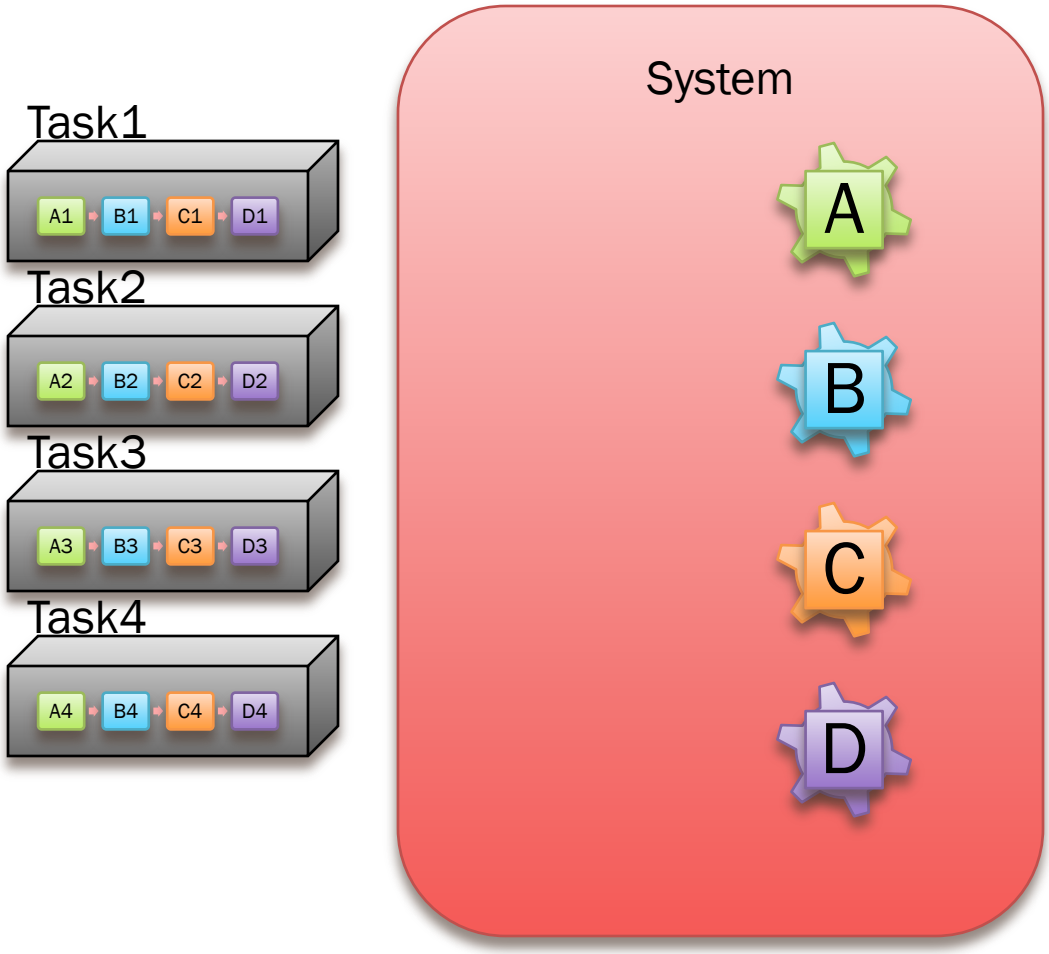
1 processing unit inside system that can undertake any step.

However, it can do the same operation on 4 pieces of data simultaneously.
(SIMD – single instruction, multiple data)

1 task takes 1 s, so throughput is 4 tasks/second

4 results appear together every second

FPGA: PIPELINING



4 processing units inside system can each undertake a different step.

Data from each task is pushed into the pipeline in sequence and moved through the processing units.

Results appear every 0.25s (after 1s). Effectively, 4 tasks/second throughput

ALTHOUGH…

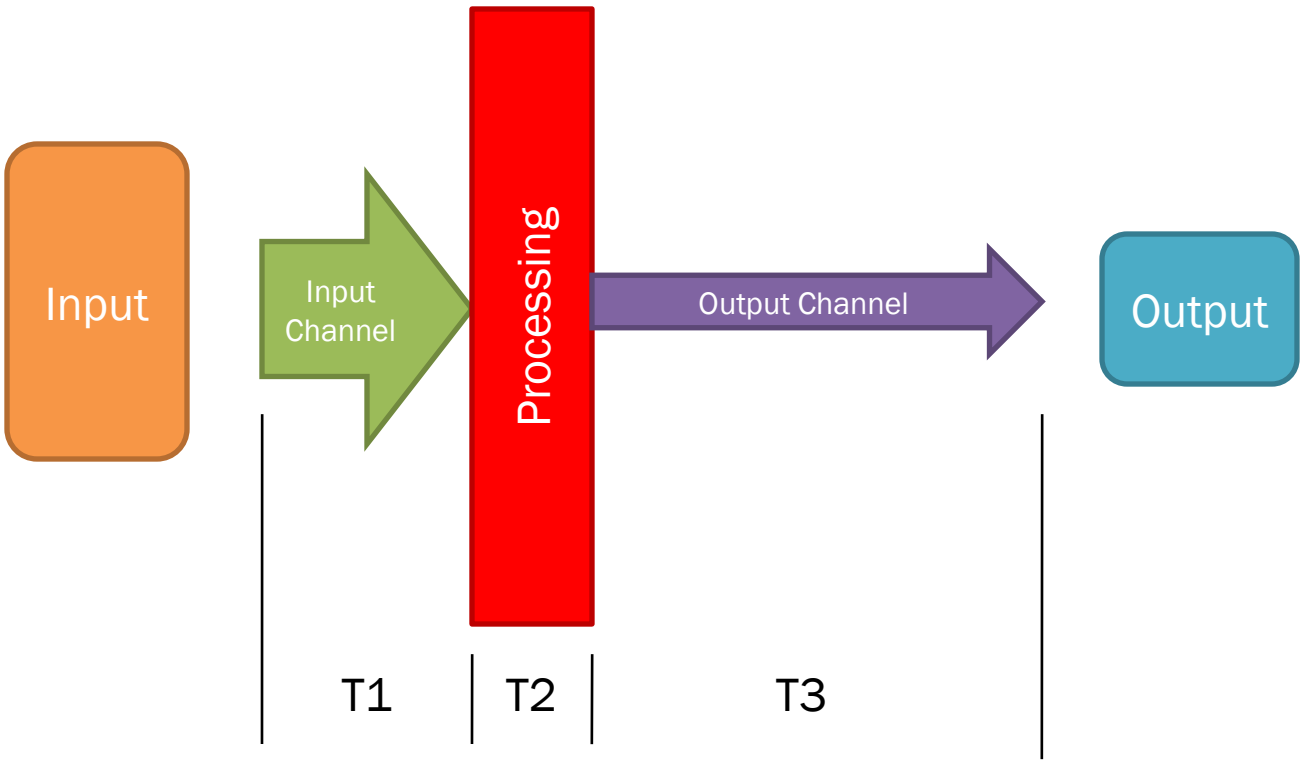
- **This is a simplification**
  - There is a great amount of difference in the internal architecture and design – but its not of interest/relevance here
- **Multi-Core CPUs have elements of pipelining in some of their operations**
  - Special compilers and coding techniques are used to access these
- **Array processors often have a level of pipelining included in their architectures**
- **FPGAs are flexible to having any combination of task-parallelism, data-parallelism or pipelining implemented**
  - But are restricted by the number of gates on the device

Processing Capacity

- **Modern CPUs operate in GHz Ranges**
  - Need to be careful with multi-core ratings. Multi-core GHz speeds are “equivalent throughput” speeds vs. Single core.
  - Each core on a multi-core runs slower than the rated GHz to preserve a heat profile
  - This means a single-threaded application can run slower on a multi-core chip than on an equivalent singe-core chip
- **Most of the exotic hardware options run at 100s MHz**
  - Array processors/GPUs, typically 200-500MHz
  - FPGAs, typically 400-800MHz
- **How large a program can be run?**
  - Exotic hardware often has restrictions on program size
  - If reconfiguration is required, this too will require being moved as data

It's not just about processing

- **Need to take into account both moving and processing data**
  - It is a mistake to purely concentrate on the processing aspects



Moving Data

- **Greatest performance cost is generally associated with moving data between 'long-term' storage & processor**
  - True at all scales – "Ye cannae break the laws o' physics"

	L-T Storage	Channel	Processor
Grid	Database or In Memory	IP Network up to switched fabric 125 Mb/s – 12 Gb/s **Typically < 200 Mb/s**	In-Memory 100s of Gb off chip L1-L3 on chip typically 8Kb-4Mb (access is still then limited by OS)
OS	In Memory	Memory buses (moving from off chip to on chip) 8 Gb/s – 50 Gb/s **Typically < 10 Gb/s**	L1-L3 on chip typically 8Kb-4Mb
Exotic H/W	In Memory	Peripheral buses (e.g. PCI-e) or specialist buses (e.g. HyperTransport) 4 Gb/s – 50 Gb/s **Typically < 5 Gb/s**	Typical exotic hardware boards have 1-256 of Gb of off-chip storage with fast banked access
Also note, highest disk access speeds are at typically <200Mb/s and at best 5Gb/s			

Advice: The only solution is to benchmark

- **There is a large amount of subtly and variation about the interplay between the communications channels, processing units and the specific algorithm**
  - Communication protocol delays
  - Choice of data transfer implementations
  - Bus choice (width, speed, etc.)
  - Motherboard design (memory, bridges, etc.)
  - OS interactions
  - Number precision (double- or single-precision)
- **Benchmarking is the only effective way to measure**
  - Vendors usually willing to provide supported investigation
  - Resist the “give us the problem and we’ll do the fast version on our hardware”. Prefer a pairing approach with your staff / contractors

SOON : HOW YOU DELIVER USING THE TECHNOLOGY

- **What is often ignored is the productivity of using technologies**
  - Often “sooner” is more important than “faster”
  - The ability to rapidly inspect and change a solution is often vital to financial applications
    - This is not necessarily so in the industries that these exotic hardware solutions come from
- **The two major factors in this are**
  - **The Programming Model**
    - How different is it to “normal” programming ?
    - What needs to be learned ? What skills are required ?
    - How long does proficiency take ?
  - **The Programming Tools**
    - How interactive are they?
    - How iterative are they? (how long does a change cycle take?)
    - What does it take to effectively debug a system?

Using Array Processors

- **Specialized array processors usually have dedicated compilers and tools**
  - Usually C-like languages with extended syntax for parallelism
    - Usually have a subset of standard C libraries
  - Edit-compile-deploy cycles – but need to have a special CPU program for integrating and uploading the "object" to the processing board, as well as communicating via the IO bus during execution
- **Example: Clearspeed**
  - Can run normal C code out-of-the-box which can be incrementally altered to be more parallel
  - Full on-chip debugging (hugely useful! And rare in other solutions)
  - Full double-precision representation (some only do single-precision)
  - Proprietary compiler but GDB compatibility
  - Card was ~£2000 in 2006 – no current data on costs (assume less now?)
  - Only 10-25W power usage, ~96 GFLOPs (~3.8 GFLOPS/W)

Using GPUs as Array Processors

- **Use either CUDA (Nvidia) or OpenCL**
  - C-like syntax with extensions for parallelism
  - Edit-compile-deploy cycle similar to other array-processors
- **Relatively primitive tools**
  - Only recently enabled on-hardware debugging – and not fully functional
  - Only available on certain OS/device combinations
- **Some can only do single-precision maths**
- **Use a lot of power and generate a lot of heat**
  - Generally more expensive in power/heat than CPUs, but greater GFLOP/W
  - ~120-900W per board (CPUs ~30-100W)
  - 500-2,100 GFLOP d.p calculation (CPU ~40-60 GFLOP)
  - 0.4-2.3 GFLOP/W (CPUs ~0.5-1.3 GFLOPS/W)
- **Large interest group and greater availability/distribution**
  - CUDA and OpenCL on freely available on Windows/Linux/Mac OSX and can work with a £600 GPU (even some built-ins)
  - More programmers available??

USING FPGAS

- **Generating FPGA 'programs' is hard**
  - Low-level VHDL or Verilog is a black-art and almost impossible to teach "real programmers"
    - Right down at the gate-level. Need to configure gates into processing units like adders, multipliers, etc (or rather configure to MLBs and specialized parts in the particular FPGA you are using)
    - NOT productive
  - SystemC : modelling language/modules for algorithmic design
  - Higher level C-like languages available
    - But they are NOT C semantics – just share the syntax
      - Very different model of programming and software
    - Have libraries and are simpler than VHDL or Verilog. Easier to teach software developers.
- **Long deployment cycle**
  - Edit-verify-compile-place&route-deploy can take hours or even DAYS!

Using FPGAs (2)

- **Primitive tools for software developers**
  - No real VHDL/Verilog interactive debugging tools
  - Higher-level tools are poor (in our experience)
    - No interactive debugging
    - Often don't have library support for more esoteric maths functions (exp!)
- **Often constrained by choice of FPGA**
  - Number of gates, types of MLB units, board configuration
    - Double-precision multiplier ~500 slices, single-precision ~130. Typical large FPGA has ~50k-100k slices
      - i.e. Can only fit ~200 d.p. Multipliers on a big FPGA
  - FPGAs are extremely specialised for different uses. Careful choice is required for specific applications
- **Very good connectivity and can deal with stream processing extremely effectively**
  - Good for doing data-stream processing or very specific low-precision calculations

What are the benefits?

- **Practical implementation and measurement of two finance examples (compute intensive)**
  - (vs. Optimized C code on 3GHz Xeon by non-vendor staff)
  - European Option Pricing (double-precision)
    - **FPGA: 2x faster** (vendor claims **16x** – possible with time)
    - **Array Processor: 92x faster**
  - CDO Pricing
    - **FPGA: 2x faster**
    - **Array Processor: 10x faster**
- **Latency reduction using FPGA as dedicated message processor**
  - Used FPGA with on-board TCP/IP stack and dedicated message translation vs. Standard computer with network card
  - **15-50x faster**

What are the challenges?

- **For FPGAs**
  - Productivity
    - Compilation can take hours – in one case took 2 days!
    - Actually learnt array processor language and got first example 4x faster during a single FPGA compile for d.p. euro option pricer
  - Programming model is non-intuitive to application programmers
- **For Array Processors**
  - General availability – still very niche
- **For GPUs as Array Processors**
  - Tools are poor
- **Software Developers need to change their approach to algorithms**
  - Not like threading or other current ‘software metaphors’
  - Conditional statements hurt!

Suggested Strategy

- **Rank your performance problems and pick the hardest or the most valuable**
- **Characterise the problems and their environment...**
  - Volume or latency?
  - Machines, network, personnel
- **...then pick your technologies**
  - Not all technologies are appropriate for all problems
- **Do a pilot benchmarking exercise on a known problem**
  - Measure system performance
  - Measure productivity
  - Use non-vendor staff (or at least a mix with your people hands-on)
  - Time-box with dedicated staff

A Final Tale

- **Matching Problem**
  - 2 x 10^{12} possible combinations
  - Need to optimize matching & profitability
- **Original Brute Force C# solution : 5.4 days**
  - Actually ran for 2 hour and covered ~2% of states
- **Optimized Brute Force C# solution : 8.6 hours (15x faster)**
  - Again 2 hours, but ~30% of states
- **GPU Brute Force solution : 1 hour (128x faster)**
  - 100% of states!
  - 1 week of programming
- **Algorithmic solution : 23 seconds (20,285x faster)**
  - 3 days of head-scratching, 1 day programming
  - Linear/Quadratic Programming

The Moral of the Tale?

- **Good algorithms usually win over fancy hardware or hand-crafted software optimization**
- **Beware the "simplest thing" approach when dealing with performance constrained problems**
  - To get it delivered sooner – simple brute force is good
  - But build it to be replaced with a good algorithm later

Other Advice

- **Get advice from people who are software people who know something about hardware – not the other way around**
  - Hardware engineers have a very different view of software and are used to working to very different set of non-functional requirements (timescales, costs, rate of change...)
- **Think about the whole system not just the technology – software, hardware, people, power, continuity...**
- **Watch Out For...**
  - "It's a hardware solution – it must be faster"
    - Not always true. Don't spend excessive time to get little return
  - "It's cool"
  - Vendor claims

QUESTIONS?

- **Any further questions ?**
- **Still interested ?**
  - What happens next ? Can I help ?

- **Email: Stigmergist@gmail.com**
- **Twitter: johnsnolan**