# High Performance Computing on the Desktop

Roman Pearce

CECM, Simon Fraser University

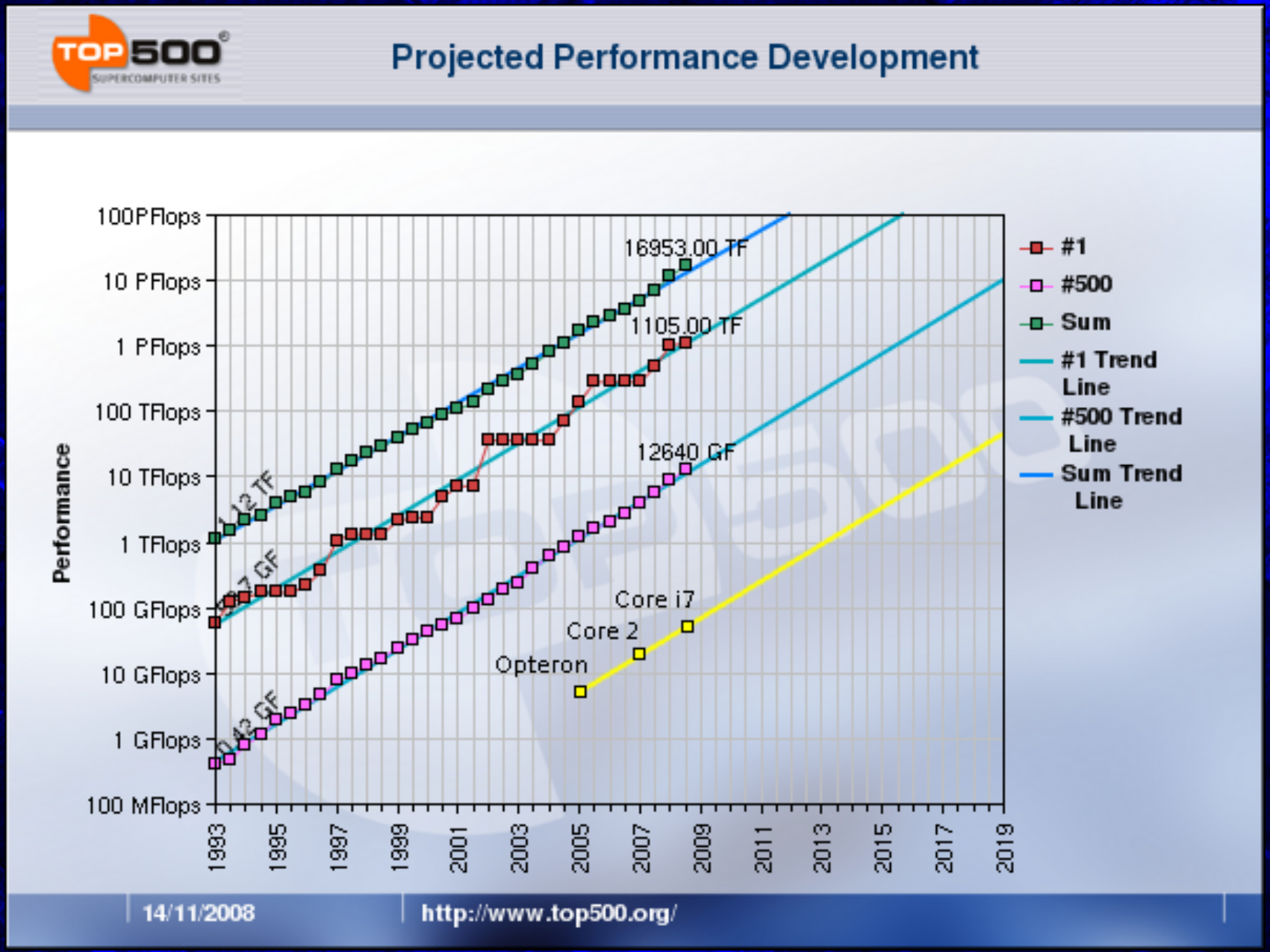
June 2009

with Michael Monagan, Simon Fraser University

# Introduction

# Introduction

# Introduction



# Introduction

How will we get there ?

- single threaded performance is at a wall
- multicore processors are widely available
- we're being pushed into parallel computing

*That is the present.*

*What about the future?*

- new types of processors
- heterogeneous computing

# Introduction

How will we get there ?

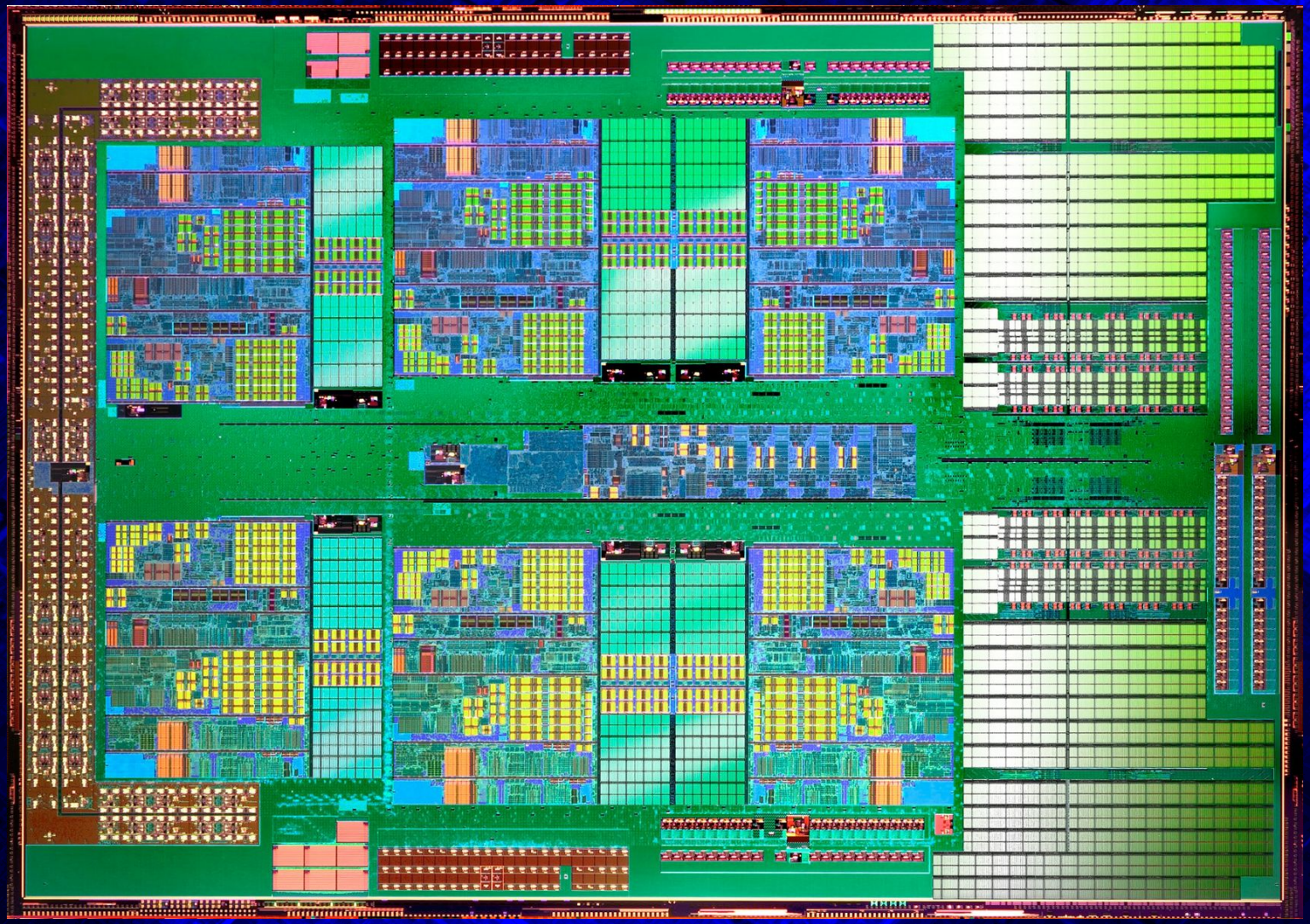
- single threaded performance is at a wall
- multicore processors are widely available
- we're being pushed into parallel computing

*That is the present.*

*What about the future?*

- new types of processors
- heterogeneous computing

# Multicore



AMD 45nm “Istanbul” – June 1, 2009

# Multicore

Today:

- dual core processors in laptops
- quad core processors in desktops
- six core processors in servers

End of 2009:

- quad core processors in laptops
- eight core processors in servers

End of 2010:

- six core processors in desktops
- twelve core processors in servers

# Multicore

Today:

- dual core processors in laptops
- quad core processors in desktops
- six core processors in servers

End of 2009:

- quad core processors in laptops
- eight core processors in servers

End of 2010:

- six core processors in desktops
- twelve core processors in servers

# Multicore

Today:

- dual core processors in laptops
- quad core processors in desktops
- six core processors in servers

End of 2009:

- quad core processors in laptops
- eight core processors in servers

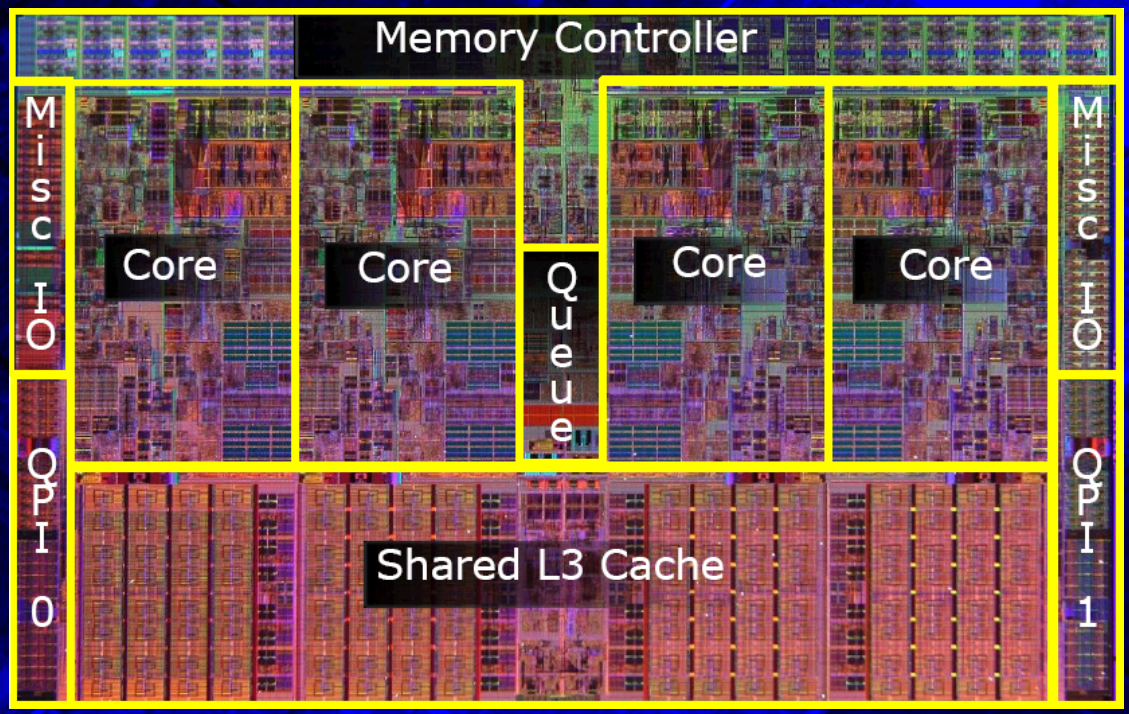
End of 2010:

- six core processors in desktops
- twelve core processors in servers

# Multicore

What can they do?

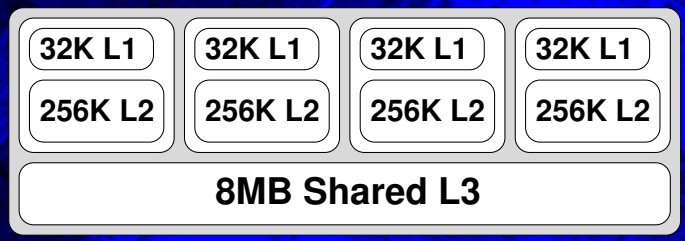
- $10^{10}$ instructions per second
- very good at branching (if ... then ... else ...)
- low latency, fast communication (shared cache)



# Multicore

*Fine-grained parallelism:*

- must be done "on the chip"
- work in cache – tight code
- use very little memory
- switch tasks – don't wait




Core i7

$$f = (1 + x + y + z + t)^{30} \quad g = f + 1$$

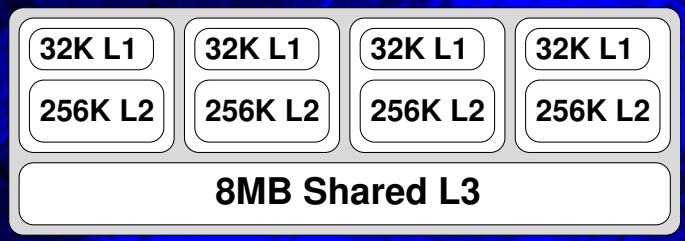
$46376 \times 46376 = 635376$ terms, $W(f, g) = 3332$

| threads | | Core i7 | | Core 2 Quad | |
|---|---|---|---|---|---|
| sdmp | 4 | 11.48 s | 6.15x | 14.15 s | 4.25x |
| | 3 | 16.63 s | 4.24x | 19.43 s | 3.10x |
| | 2 | 28.26 s | 2.50x | 28.29 s | 2.13x |
| | 1 | 70.59 s | | 60.25 s | |
| Magma 2.15-8 | 1 | 526.12 s | | | |
| Pari/GP 2.3.3 | 1 | 642.74 s | | 707.61 s | |
| Singular 3-1-0 | 1 | 744.00 s | | 1048.00 s | |
| Maple 13 | 1 | 5849.48 s | | 9343.68 s | |

# Multicore

*Fine-grained parallelism:*

- must be done "on the chip"
- work in cache – tight code
- use very little memory
- switch tasks – don't wait




Core i7

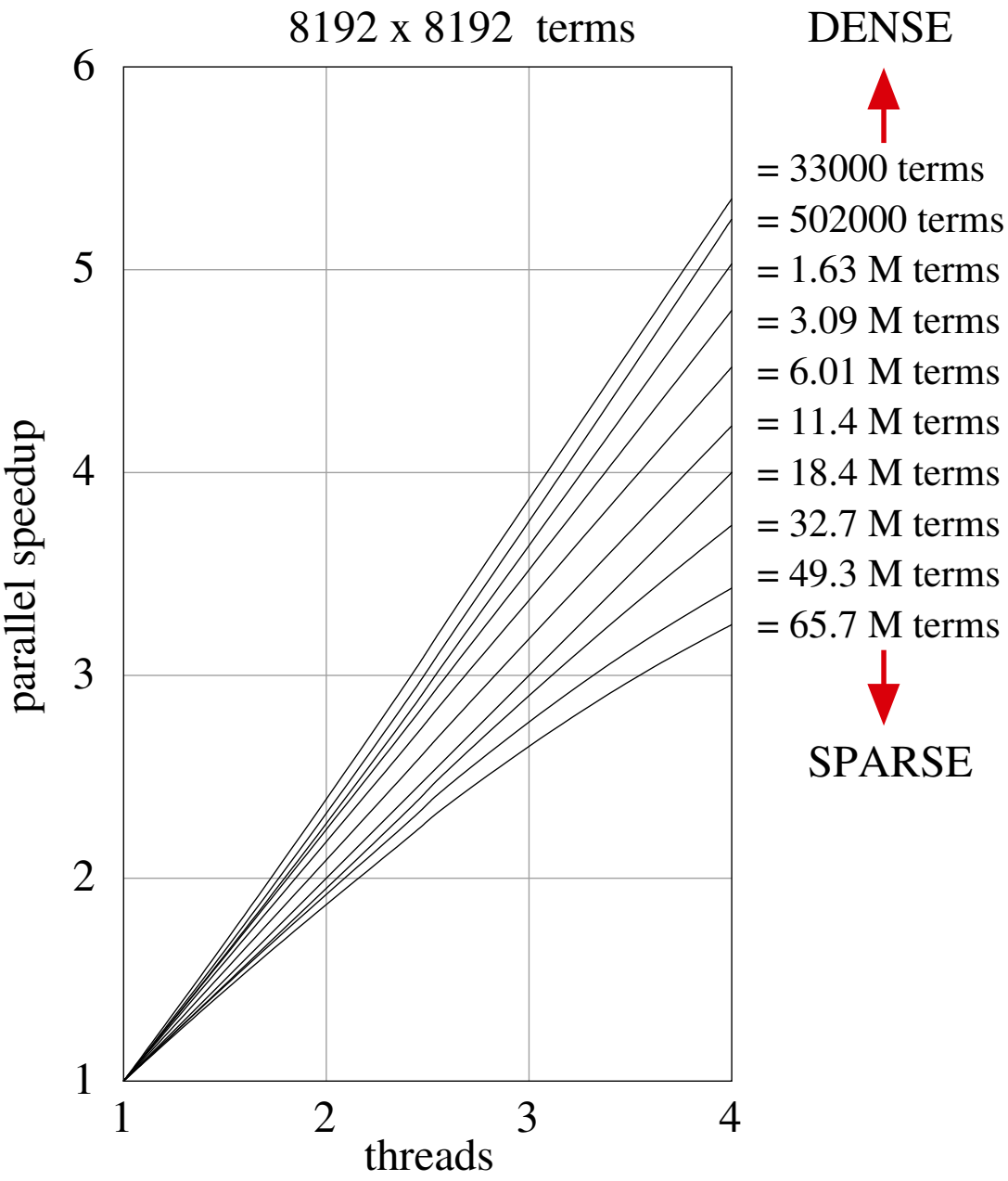
$$f = (1 + x + y + z + t)^{30} \quad g = f + 1$$

$46376 \times 46376 = 635376$ terms, $W(f, g) = 3332$

| | threads | Core i7 | | Core 2 Quad | |
|---|---|---|---|---|---|
| sdmp | 4 | 11.48 s | 6.15x | 14.15 s | 4.25x |
| | 3 | 16.63 s | 4.24x | 19.43 s | 3.10x |
| | 2 | 28.26 s | 2.50x | 28.29 s | 2.13x |
| | 1 | 70.59 s | | 60.25 s | |
| Magma 2.15-8 | 1 | 526.12 s | | | |
| Pari/GP 2.3.3 | 1 | 642.74 s | | 707.61 s | |
| Singular 3-1-0 | 1 | 744.00 s | | 1048.00 s | |
| Maple 13 | 1 | 5849.48 s | | 9343.68 s | |

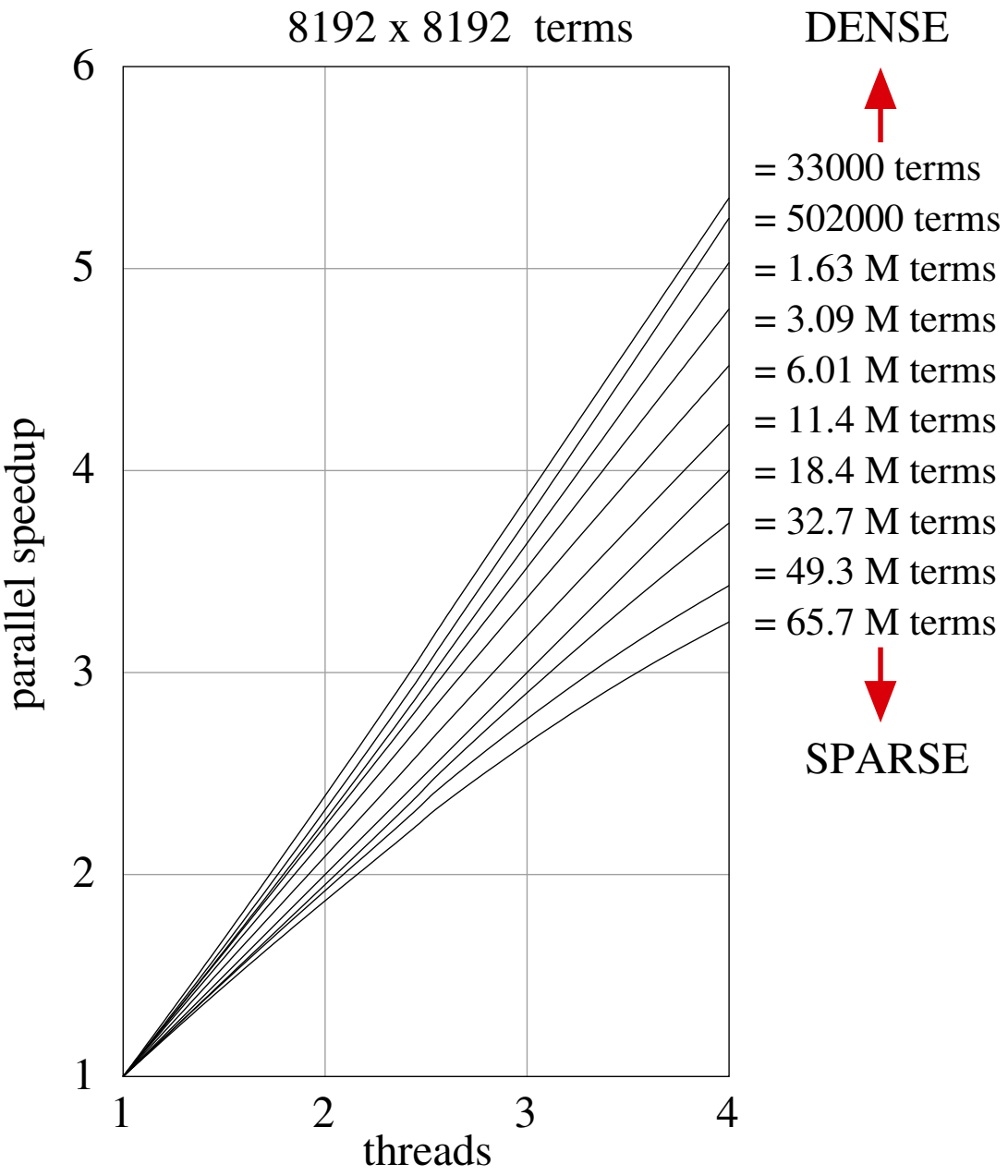
# Multicore

- linear speedup is a realistic goal
- even for sparse problems
- often superlinear in practice



# Multicore

- linear speedup is a realistic goal
- even for sparse problems
- often superlinear in practice



# Multicore

| ⟵ efficiency | parallel speedup ⟶ | | | | | |
|---|---|---|---|---|---|---|
| | 1.7 | 2.9 | 4.9 | 8.4 | 14.2 | 24.1 |
| | 1.8 | 3.2 | 5.8 | 10.5 | 18.9 | 34.0 |
| | 1.9 | 3.6 | 6.8 | 13.0 | 24.7 | 47.0 |
| | 2.0 | 4.0 | 8.0 | 16.0 | 32.0 | 64.0 |
| | 2.1 | 4.4 | 9.3 | 19.5 | 40.8 | 85.7 |
| | 2.2 | 4.8 | 10.6 | 23.4 | 51.5 | 113.4 |
| | 2.3 | 5.3 | 12.2 | 28.0 | 64.4 | 148.0 |
| | 2.4 | 5.7 | 13.8 | 33.2 | 79.6 | 191.1 |
| | 2.5 | 6.2 | 15.6 | 39.0 | 97.6 | 244.1 |

Optimization pays off.

- sequential code: $10 - 100x$ faster
- parallel code: $50 - 10000x$ faster

Challenge: *quasi-linear time algorithms*

# Multicore

| ⟵ efficiency | parallel speedup ⟶ | | | | | |
|---|---|---|---|---|---|---|
| | 1.7 | 2.9 | 4.9 | 8.4 | 14.2 | 24.1 |
| | 1.8 | 3.2 | 5.8 | 10.5 | 18.9 | 34.0 |
| | 1.9 | 3.6 | 6.8 | 13.0 | 24.7 | 47.0 |
| | 2.0 | 4.0 | 8.0 | 16.0 | 32.0 | 64.0 |
| | 2.1 | 4.4 | 9.3 | 19.5 | 40.8 | 85.7 |
| | 2.2 | 4.8 | 10.6 | 23.4 | 51.5 | 113.4 |
| | 2.3 | 5.3 | 12.2 | 28.0 | 64.4 | 148.0 |
| | 2.4 | 5.7 | 13.8 | 33.2 | 79.6 | 191.1 |
| | 2.5 | 6.2 | 15.6 | 39.0 | 97.6 | 244.1 |

Optimization pays off.

- sequential code: $10 - 100x$ faster
- parallel code: $50 - 10000x$ faster

Challenge: *quasi-linear time algorithms*

# Multicore

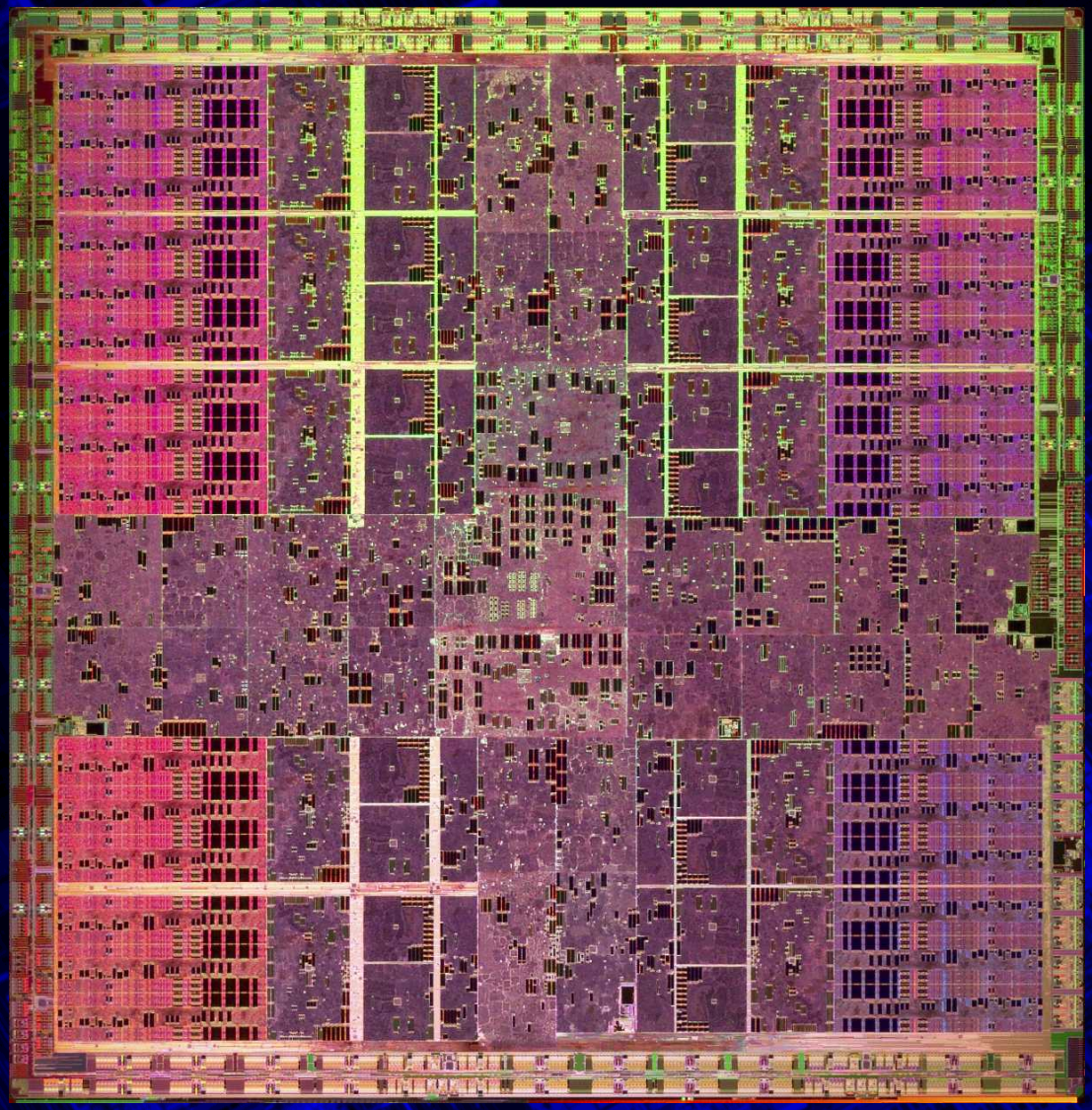
*Fine-grained parallelism.*

*Fast and nimble algorithms.*

*Large structures that can't be broken up.*

- sparse polynomials
- sparse linear algebra
- sparse graphs and networks
- integer computations

# Graphics Processors



Nvidia GT200

# Graphics Processors

Today:

- GPUs in some laptops
- $100 – 500 add in card for desktops
- 4 to 8 cards in compute servers
- Nvidia's CUDA

End of 2009:

- cross platform, cross vendor library: OpenCL
- flood of development

End of 2010:

- CPU + GPUs on a single chip

# Graphics Processors

Today:

- GPUs in some laptops
- $100 – 500 add in card for desktops
- 4 to 8 cards in compute servers
- Nvidia's CUDA

End of 2009:

- cross platform, cross vendor library: OpenCL
- flood of development

End of 2010:

- CPU + GPUs on a single chip

# Graphics Processors

Today:

- GPUs in some laptops
- $100 – 500 add in card for desktops
- 4 to 8 cards in compute servers
- Nvidia's CUDA

End of 2009:

- cross platform, cross vendor library: OpenCL
- flood of development

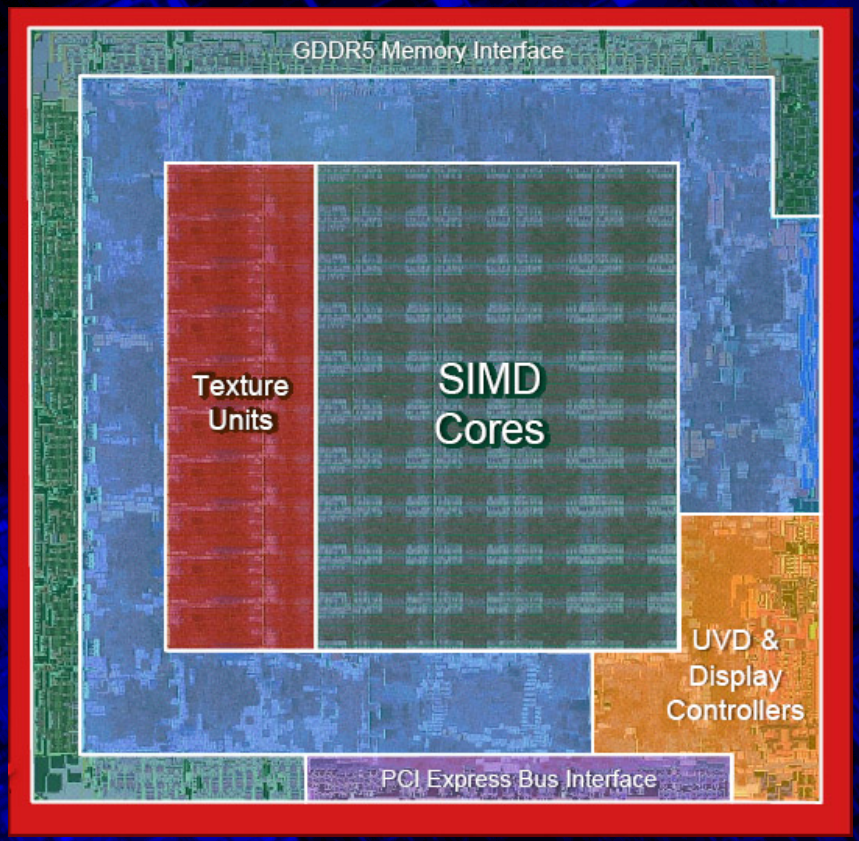
End of 2010:

- CPU + GPUs on a single chip

# Graphics Processors

What can they do?

- $10^{12}$ FLOPS (single precision)
- incredible bandwidth (150 GB/sec)
- uniform execution across cores



# Graphics Processors

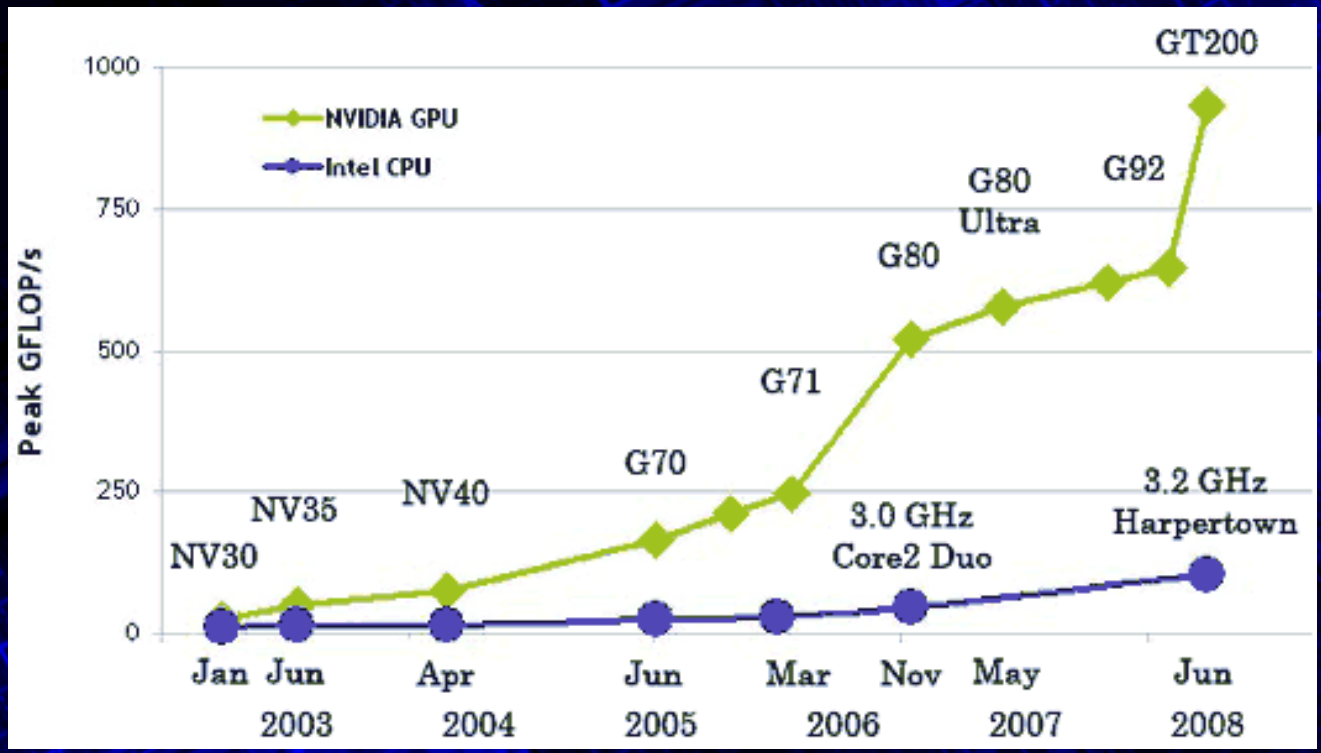
*Data parallelism:*

- process small blocks of data independently
- simple operations, massive parallelism
- create thousands of threads
- rely on throughput

# Graphics Processors

*Data parallelism:*

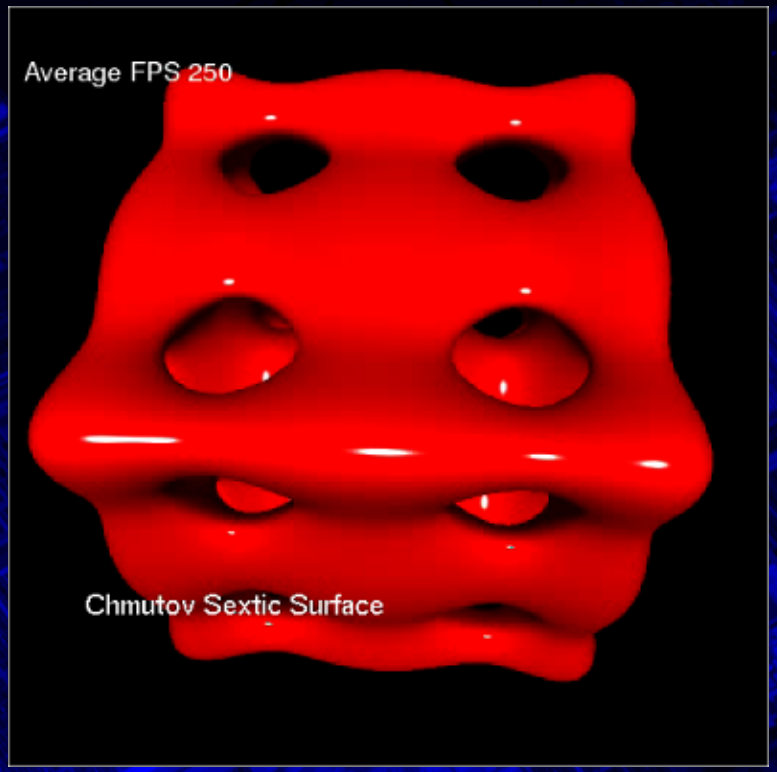
- process small blocks of data independently
- simple operations, massive parallelism
- create thousands of threads
- rely on throughput



# Graphics Processors

## Visualization

- Adaptive Marching Points (Singh & Narayanan 2008)
- Real Time Ray Tracing (Reimers & Seland 2008)




## Industrial Linear Systems

- $163840 \times 163840$ dense
  2.5 hrs $\longrightarrow$ 5.5 min
  (Ibragimov 2009)

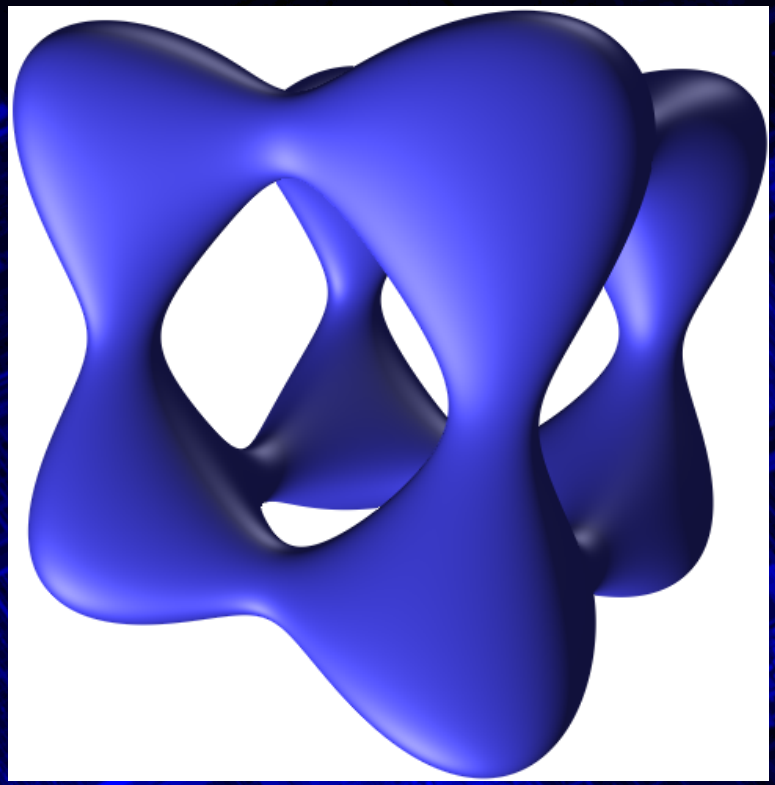
## Simulations

- Stochastic Differential Equations (Januszewski & Kostur 2009)

# Graphics Processors

Visualization

- Adaptive Marching Points (Singh & Narayanan 2008)
- Real Time Ray Tracing (Reimers & Seland 2008)



Industrial Linear Systems

- $163840 \times 163840$ dense
  2.5 hrs $\longrightarrow$ 5.5 min
  (Ibragimov 2009)

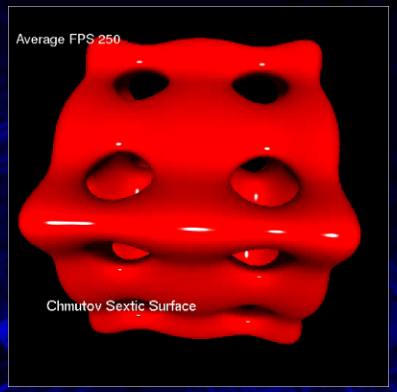
Simulations

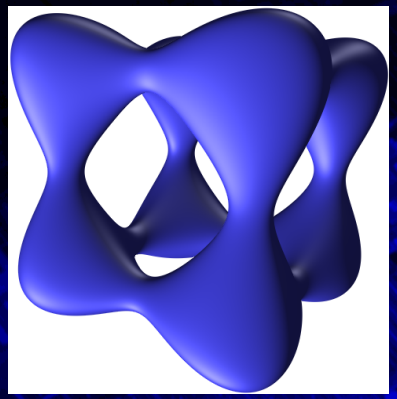
- Stochastic Differential Equations (Januszewski & Kostur 2009)

# Graphics Processors

## Visualization

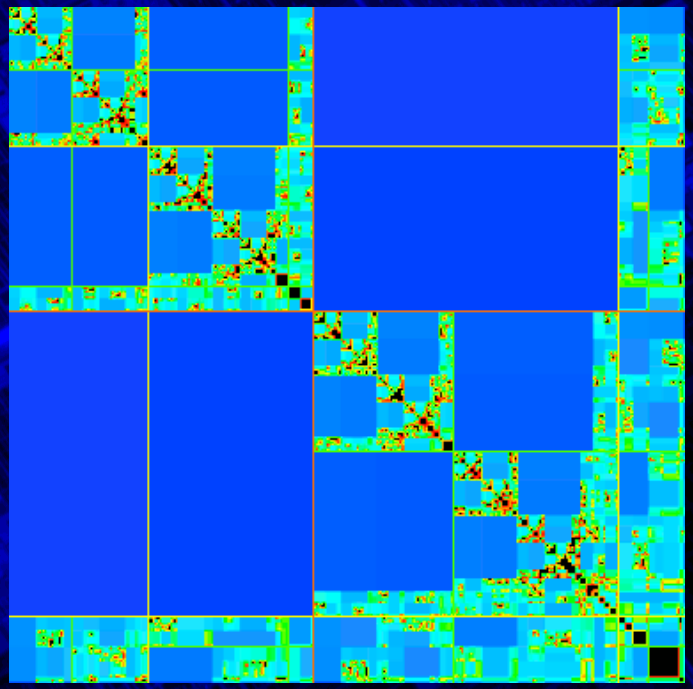
- Adaptive Marching Points (Singh & Narayanan 2008)
- Real Time Ray Tracing (Reimers & Seland 2008)



## Industrial Linear Systems

- $163840 \times 163840$ dense
  2.5 hrs $\longrightarrow$ 5.5 min
  (Ibragimov 2009)



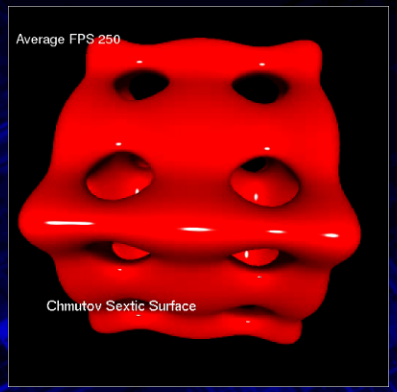
## Simulations

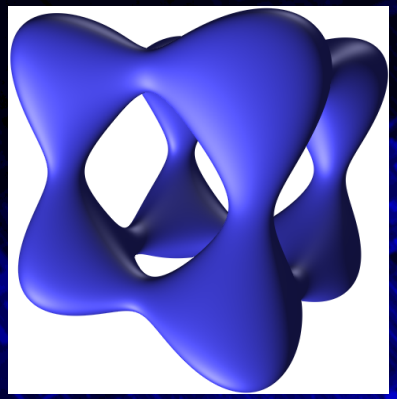
- Stochastic Differential Equations (Januszewski & Kostur 2009)

# Graphics Processors

## Visualization

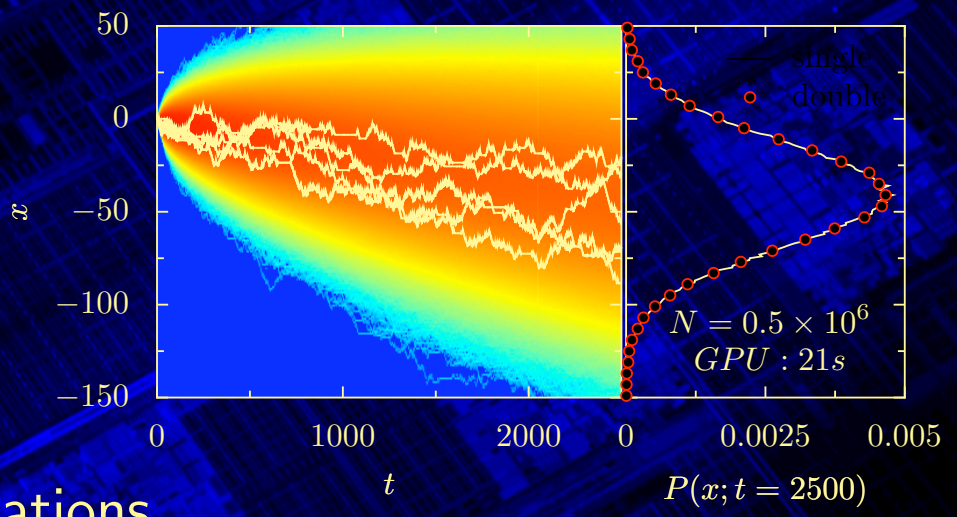
- Adaptive Marching Points (Singh & Narayanan 2008)
- Real Time Ray Tracing (Reimers & Seland 2008)



## Industrial Linear Systems

- $163840 \times 163840$ dense
  2.5 hrs $\longrightarrow$ 5.5 min
  (Ibragimov 2009)




## Simulations

- Stochastic Differential Equations (Januszewski & Kostur 2009)

# Graphics Processors

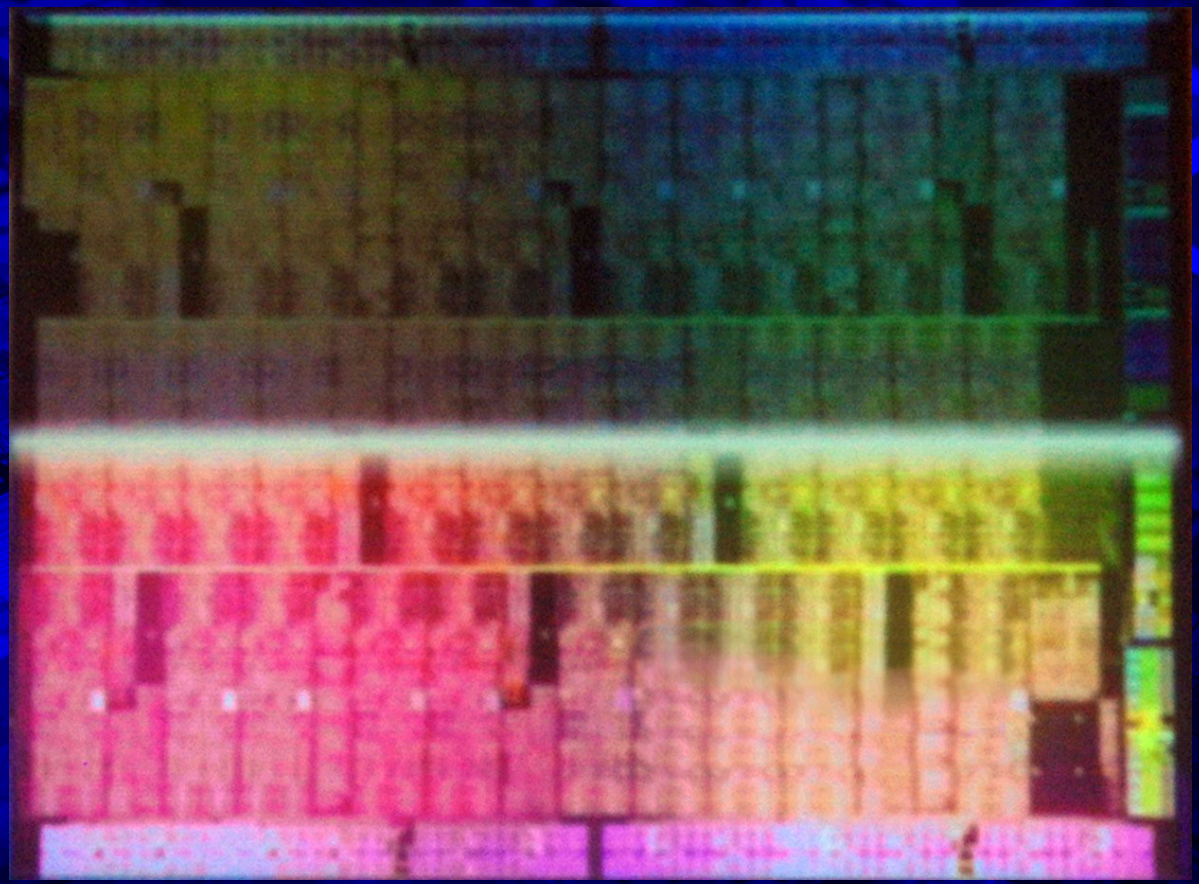
*Uniform computations.*

*Massive throughput.*

*Dense structures.*

- dense polynomials
- dense linear algebra
- dense graphs and networks
- signal processing
- visualization
- simulation
- etc.

# Larrabee



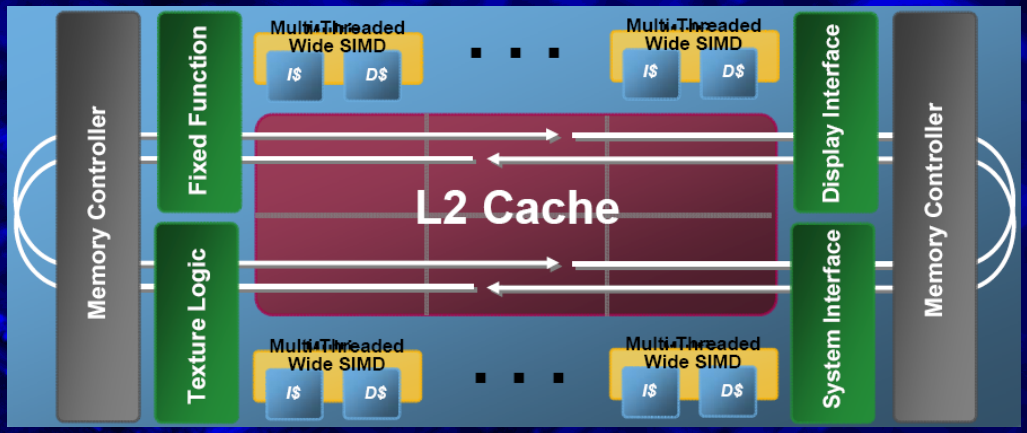
Intel “Larrabee”

# Larrabee

Coming in early 2010.

Many-core x86:

- 32 cores initially
- 4 threads per core
- 512-bit vector units
- $10^{12}$ FLOPS at 2GHz




*Fully programmable "graphics" processor.*

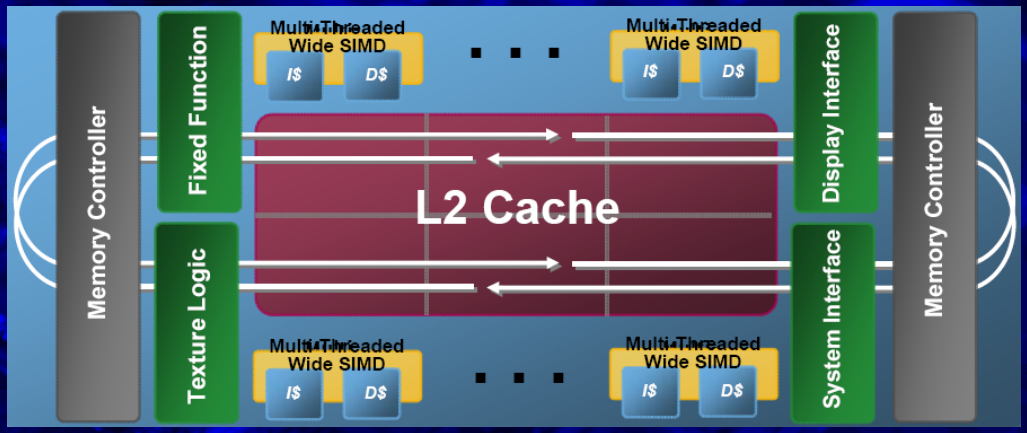
- coherent shared cache
- supports recursion

# Larrabee

Coming in early 2010.

Many-core x86:

- 32 cores initially
- 4 threads per core
- 512-bit vector units
- $10^{12}$ FLOPS at 2GHz




*Fully programmable "graphics" processor.*

- coherent shared cache
- supports recursion

# Larrabee

*Divide-and-conquer.*

*Adaptive algorithms.*

*High throughput applications.*

- numerical solving
- numerical integration
- sparse linear algebra
- visualization
- simulation
- etc.

Questions?