A computer algebraist meets a computer centre director

James Davenport
Hebron & Medlock Professor of Information Technology
Chairman, Powerful Computing Working Party

University of Bath (U.K.)
(visiting Waterloo)

25 June 2009

Many thanks to all at Bath, and Prof. Guest (Cardiff)

University of Bath

University of Bath

- Good (9th out of 117 in the U.K.: Guardian 12 May 2009)
- Heavily co-op
- Strengths in Science, Engineering, Mathematics

University of Bath

- Good (9th out of 117 in the U.K.: Guardian 12 May 2009)
- Heavily co-op
- Strengths in Science, Engineering, Mathematics

But small — 538 Faculty

Bath recently bought

Bath recently bought

- 100 node machine

Bath recently bought

- 100 node machine
- Each node 2×quad-core Intel ‘Harpertown’; 16GB main memory

Bath recently bought

- 100 node machine
- Each node 2×quad-core Intel ‘Harpertown’; 16GB main memory
- Interconnect with Gigabit Ethernet and DDR Infiniband

Bath recently bought

- 100 node machine
- Each node 2×quad-core Intel 'Harpertown'; 16GB main memory
- Interconnect with Gigabit Ethernet and DDR Infiniband

* The Infiniband cost 20% of the total, so we *want* to run communicating multi-node jobs.

Bath recently bought

- 100 node machine
- Each node 2×quad-core Intel 'Harpertown'; 16GB main memory
- Interconnect with Gigabit Ethernet and DDR Infiniband
* The Infiniband cost 20% of the total, so we *want* to run communicating multi-node jobs.
- 7 Linpack Tflop/sec.

Bath recently bought

- 100 node machine
- Each node 2×quad-core Intel 'Harpertown'; 16GB main memory
- Interconnect with Gigabit Ethernet and DDR Infiniband
* The Infiniband cost 20% of the total, so we *want* to run communicating multi-node jobs.
- 7 Linpack Tflop/sec.
- Therefore about 2,000th in the world.

Bath recently bought

- 100 node machine
- Each node 2×quad-core Intel 'Harpertown'; 16GB main memory
- Interconnect with Gigabit Ethernet and DDR Infiniband
* The Infiniband cost 20% of the total, so we *want* to run communicating multi-node jobs.
- 7 Linpack Tflop/sec.
- Therefore about 2,000th in the world.

Bath recently bought

- 100 node machine
- Each node 2×quad-core Intel 'Harpertown'; 16GB main memory
- Interconnect with Gigabit Ethernet and DDR Infiniband
* The Infiniband cost 20% of the total, so we *want* to run communicating multi-node jobs.
- 7 Linpack Tflop/sec.
- Therefore about 2,000th in the world.

To do "capacity" computing, not necessarily "capability"

Operates as a batch service

Operates as a batch service

- The unit of allocation is the node

Operates as a batch service

- The unit of allocation is the node
  (8 cores, 16GB memory, 70 Gflops)

Operates as a batch service

- The unit of allocation is the node
  (8 cores, 16GB memory, 70 Gflops)
- Nodes rebooted between uses

Operates as a batch service

- The unit of allocation is the node
  (8 cores, 16GB memory, 70 Gflops)
- Nodes rebooted between uses
- Maximum duration 48 hours

Operates as a batch service

- The unit of allocation is the node

  (8 cores, 16GB memory, 70 Gflops)
- Nodes rebooted between uses
- Maximum duration 48 hours
- Very few applications use more than 10–20 nodes

Operates as a batch service

- The unit of allocation is the node
  (8 cores, 16GB memory, 70 Gflops)
- Nodes rebooted between uses
- Maximum duration 48 hours
- Very few applications use more than 10–20 nodes
- **No** interactivity

Operates as a batch service

- The unit of allocation is the node
  (8 cores, 16GB memory, 70 Gflops)
- Nodes rebooted between uses
- Maximum duration 48 hours
- Very few applications use more than 10–20 nodes
- **No** interactivity

* Some services with special graphics do offer interactivity.

Three levels of Portability

Three levels of Portability

1. Will my program run on that machine?

Three levels of Portability

1. Will my program run on that machine?
2. Will my program run on that machine

Three levels of Portability

1. Will my program run on that machine?
2. Will my program run on that machine

Three levels of Portability

1. Will my program run on that machine?
2. Will my program run on that machine and give the same results?

Three levels of Portability

1. Will my program run on that machine?
2. Will my program run on that machine and give the same results?

* Language standards; IEEE etc.

Three levels of Portability

1. Will my program run on that machine?
2. Will my program run on that machine and give the same results?

   * Language standards; IEEE etc.

3. Will my program run on that machine and give the same results,

Three levels of Portability

1. Will my program run on that machine?
2. Will my program run on that machine and give the same results?

   * Language standards; IEEE etc.

3. Will my program run on that machine and give the same results,

Three levels of Portability

1. Will my program run on that machine?
2. Will my program run on that machine and give the same results?
   * Language standards; IEEE etc.
3. Will my program run on that machine and give the same results, and do so efficiently?
   * Still an unsolved problem.

BLAS — Basic Linear Algebra Subprograms

A tale of three levels.

BLAS — Basic Linear Algebra Subprograms

A tale of three levels.

1. Vector–Vector operations

BLAS — Basic Linear Algebra Subprograms

A tale of three levels.

1. Vector–Vector operations
   * Great for the Cray-1, and other vector processors.

BLAS — Basic Linear Algebra Subprograms

A tale of three levels.

1. Vector–Vector operations
   * Great for the Cray-1, and other vector processors.
2. Matrix–Vector operations

BLAS — Basic Linear Algebra Subprograms

A tale of three levels.

1. Vector–Vector operations
   * Great for the Cray-1, and other vector processors.
2. Matrix–Vector operations
   * Great for SIMD machines

BLAS — Basic Linear Algebra Subprograms

A tale of three levels.

1. Vector–Vector operations
   * Great for the Cray-1, and other vector processors.
2. Matrix–Vector operations
   * Great for SIMD machines
3. Matrix–Matrix operations

BLAS — Basic Linear Algebra Subprograms

A tale of three levels.

1. Vector–Vector operations
   * Great for the Cray-1, and other vector processors.
2. Matrix–Vector operations
   * Great for SIMD machines
3. Matrix–Matrix operations
   * By doing $O(n^3)$ operations on $O(n^2)$ objects, they make effective use of the cache(s).

BLAS — Basic Linear Algebra Subprograms

A tale of three levels.

1. Vector–Vector operations
   * Great for the Cray-1, and other vector processors.
2. Matrix–Vector operations
   * Great for SIMD machines
3. Matrix–Matrix operations
   * By doing $O(n^3)$ operations on $O(n^2)$ objects, they make effective use of the cache(s).
   * Very heavily tuned, manually or automatically, to the machine.

BLAS — Basic Linear Algebra Subprograms

A tale of three levels.

1. Vector–Vector operations
   * Great for the Cray-1, and other vector processors.
2. Matrix–Vector operations
   * Great for SIMD machines
3. Matrix–Matrix operations
   * By doing $O(n^3)$ operations on $O(n^2)$ objects, they make effective use of the cache(s).
   * Very heavily tuned, manually or automatically, to the machine.

BLAS — Basic Linear Algebra Subprograms

A tale of three levels.

1. Vector–Vector operations
   * Great for the Cray-1, and other vector processors.
2. Matrix–Vector operations
   * Great for SIMD machines
3. Matrix–Matrix operations
   * By doing $O(n^3)$ operations on $O(n^2)$ objects, they make effective use of the cache(s).
   * Very heavily tuned, manually or automatically, to the machine.

What does computer algebra have that's similar?

Moore's Law *is* Still Valid

- The number of transistors on a chip *is* still roughly doubling every 18 months

Moore's Law *is* Still Valid

- The number of transistors on a chip *is* still roughly doubling every 18 months
- **But** this is not translating into faster scalar performance.

Moore's Law *is* Still Valid

- The number of transistors on a chip *is* still roughly doubling every 18 months
- **But** this is not translating into faster scalar performance.
- Instead, we get more cores, more and larger caches/ Translation Lookaside Buffers, and more 'features':

Moore's Law *is* Still Valid

- The number of transistors on a chip *is* still roughly doubling every 18 months
- **But** this is not translating into faster scalar performance.
- Instead, we get more cores, more and larger caches/ Translation Lookaside Buffers, and more 'features':

RSQRTPS Compute Reciprocals of Square Roots of Packed Single-Precision Floating-Point Values

Moore's Law *is* Still Valid

- The number of transistors on a chip *is* still roughly doubling every 18 months
- **But** this is not translating into faster scalar performance.
- Instead, we get more cores, more and larger caches/ Translation Lookaside Buffers, and more 'features':

RSQRTPS Compute Reciprocals of Square Roots of Packed Single-Precision Floating-Point Values

Moore's Law *is* Still Valid

- The number of transistors on a chip *is* still roughly doubling every 18 months
- **But** this is not translating into faster scalar performance.
- Instead, we get more cores, more and larger caches/ Translation Lookaside Buffers, and more 'features':

RSQRTPS Compute Reciprocals of Square Roots of Packed Single-Precision Floating-Point Values

"These guys have more silicon than they know what to do with"

Moore's Law *is* Still Valid (II)

Moore's Law *is* Still Valid (II)

The Istanbul Opterons contain 904 million transistors, which consist of six cores, each with 64 KB of L1 data cache, 64 KB of L1 instruction cache, and 512 KB of L2 cache per core. . . . Each chip also has 6 MB of L3 cache that is shared by all of the cores

Moore's Law *is* Still Valid (II)

The Istanbul Opterons contain 904 million transistors, which consist of six cores, each with 64 KB of L1 data cache, 64 KB of L1 instruction cache, and 512 KB of L2 cache per core. . . . Each chip also has 6 MB of L3 cache that is shared by all of the cores

"I'd hate to have to implement the BLAS on that!"

Moore's Law *is* Still Valid (II)

The Istanbul Opterons contain 904 million transistors, which consist of six cores, each with 64 KB of L1 data cache, 64 KB of L1 instruction cache, and 512 KB of L2 cache per core. . . . Each chip also has 6 MB of L3 cache that is shared by all of the cores
"I'd hate to have to implement the BLAS on that!"
"That's actually not very much cache for the compute power".

Levels of parallelism

Levels of parallelism

- Within a core: basically a SIMD machine, e.g. 16 bytes at a time;

Levels of parallelism

- Within a core: basically a SIMD machine, e.g. 16 bytes at a time;
- Nehalem — two threads/core

Levels of parallelism

- Within a core: basically a SIMD machine, e.g. 16 bytes at a time;
- Nehalem — two threads/core
- multiple cores/chip (4 currently . . .)

Levels of parallelism

- Within a core: basically a SIMD machine, e.g. 16 bytes at a time;
- Nehalem — two threads/core
- multiple cores/chip (4 currently . . .)
* (shared and non-shared caches)

Levels of parallelism

- Within a core: basically a SIMD machine, e.g. 16 bytes at a time;
- Nehalem — two threads/core
- multiple cores/chip (4 currently . . .)
* (shared and non-shared caches)
- multiple chips/node (often 2, but 4 is happening)

Levels of parallelism

- Within a core: basically a SIMD machine, e.g. 16 bytes at a time;
- Nehalem — two threads/core
- multiple cores/chip (4 currently . . .)
* (shared and non-shared caches)
- multiple chips/node (often 2, but 4 is happening)
* shared memory

Levels of parallelism

- Within a core: basically a SIMD machine, e.g. 16 bytes at a time;
- Nehalem — two threads/core
- multiple cores/chip (4 currently . . .)
* (shared and non-shared caches)
- multiple chips/node (often 2, but 4 is happening)
* shared memory
- multiple nodes/machine

Levels of parallelism

- Within a core: basically a SIMD machine, e.g. 16 bytes at a time;
- Nehalem — two threads/core
- multiple cores/chip (4 currently . . .)
* (shared and non-shared caches)
- multiple chips/node (often 2, but 4 is happening)
* shared memory
- multiple nodes/machine
* How are they connected?

Levels of parallelism

- Within a core: basically a SIMD machine, e.g. 16 bytes at a time;
- Nehalem — two threads/core
- multiple cores/chip (4 currently . . .)
* (shared and non-shared caches)
- multiple chips/node (often 2, but 4 is happening)
* shared memory
- multiple nodes/machine
* How are they connected?

Levels of parallelism

- Within a core: basically a SIMD machine, e.g. 16 bytes at a time;
- Nehalem — two threads/core
- multiple cores/chip (4 currently . . .)
  * (shared and non-shared caches)
- multiple chips/node (often 2, but 4 is happening)
  * shared memory
- multiple nodes/machine
  * How are they connected?

Which level(s) are we using?

Levels of parallelism

- Within a core: basically a SIMD machine, e.g. 16 bytes at a time;
- Nehalem — two threads/core
- multiple cores/chip (4 currently . . .)
  * (shared and non-shared caches)
- multiple chips/node (often 2, but 4 is happening)
  * shared memory
- multiple nodes/machine
  * How are they connected?

Which level(s) are we using?
Which level(s) can computer algebra use?

Can we make efficient use of a single core?

- Some non-f.p. applications do.

Can we make efficient use of a single core?

- Some non-f.p. applications do.
- Skein (A SHA-3 contender) uses 6.1 cycles/byte, despite the fact that 72 rounds are being performed.

Can we make efficient use of a single core?

- Some non-f.p. applications do.
- Skein (A SHA-3 contender) uses 6.1 cycles/byte, despite the fact that 72 rounds are being performed.
- Therefore they are getting *at least* 12-byte wide parallelism.

Can we make efficient use of a single core?

- Some non-f.p. applications do.
- Skein (A SHA-3 contender) uses 6.1 cycles/byte, despite the fact that 72 rounds are being performed.
- Therefore they are getting *at least* 12-byte wide parallelism.
- GMP etc. probably get good performance *if the operands are large enough*.

Can we make efficient use of a single core?

- Some non-f.p. applications do.
- Skein (A SHA-3 contender) uses 6.1 cycles/byte, despite the fact that 72 rounds are being performed.
- Therefore they are getting *at least* 12-byte wide parallelism.
- GMP etc. probably get good performance *if the operands are large enough*.

Can we make efficient use of a single core?

- Some non-f.p. applications do.
- Skein (A SHA-3 contender) uses 6.1 cycles/byte, despite the fact that 72 rounds are being performed.
- Therefore they are getting *at least* 12-byte wide parallelism.
- GMP etc. probably get good performance *if the operands are large enough*.

But in general, how much do we have that is that fine-grained SIMD-like parallel?

Can we make efficient use of a single chip?

- want to use commonality of memory, and some commonality of cache.

Can we make efficient use of a single chip?

- want to use commonality of memory, and some commonality of cache.
- Maybe work modulo several primes in parallel

Can we make efficient use of a single chip?

- want to use commonality of memory, and some commonality of cache.
- Maybe work modulo several primes in parallel
* But if the running time depends on the prime (e.g. Cantor-Zassenhaus) we are working at the speed of the slowest.

Can we make efficient use of a single chip?

- want to use commonality of memory, and some commonality of cache.
- Maybe work modulo several primes in parallel
* But if the running time depends on the prime (e.g. Cantor-Zassenhaus) we are working at the speed of the slowest.
- Maybe try several critical pairs in Buchberger's algorithm in parallel.

Can we make efficient use of a single chip?

- want to use commonality of memory, and some commonality of cache.
- Maybe work modulo several primes in parallel
* But if the running time depends on the prime (e.g. Cantor-Zassenhaus) we are working at the speed of the slowest.
- Maybe try several critical pairs in Buchberger's algorithm in parallel.
- Serious issues of memory management/garbage collection.

Can we make efficient use of a single chip?

- want to use commonality of memory, and some commonality of cache.
- Maybe work modulo several primes in parallel
* But if the running time depends on the prime (e.g. Cantor-Zassenhaus) we are working at the speed of the slowest.
- Maybe try several critical pairs in Buchberger's algorithm in parallel.
- Serious issues of memory management/garbage collection.

Can we make efficient use of a single chip?

- want to use commonality of memory, and some commonality of cache.
- Maybe work modulo several primes in parallel
* But if the running time depends on the prime (e.g. Cantor-Zassenhaus) we are working at the speed of the slowest.
- Maybe try several critical pairs in Buchberger's algorithm in parallel.
- Serious issues of memory management/garbage collection.

It's not clear that we *do*: can we?

Can we make efficient use of a multiple nodes? (Even if they were available)

- As far as possible, overlap computation and communication.

Can we make efficient use of a multiple nodes? (Even if they were available)

- As far as possible, overlap computation and communication.
- Try different orders (in Gröbner, CAD etc.) in parallel.

Can we make efficient use of a multiple nodes? (Even if they were available)

- As far as possible, overlap computation and communication.
- Try different orders (in Gröbner, CAD etc.) in parallel.
- If we have a 'race' there is no issue of merging the data form different systems

The Director concludes

The Director concludes

- There's not much point in my giving you several nodes if you can't demonstrate that you're making efficient use of one!

The Director concludes

- There's not much point in my giving you several nodes if you can't demonstrate that you're making efficient use of one!
- I'm currently set up for very regular problems — yours seem to fit more into a "cloud computing" model than my service's.

The Director concludes

- There's not much point in my giving you several nodes if you can't demonstrate that you're making efficient use of one!
- I'm currently set up for very regular problems — yours seem to fit more into a "cloud computing" model than my service's.
- Don't be too depressed — I said that to the MatLab people as well.

The Director concludes

- There's not much point in my giving you several nodes if you can't demonstrate that you're making efficient use of one!
- I'm currently set up for very regular problems — yours seem to fit more into a "cloud computing" model than my service's.
- Don't be too depressed — I said that to the MatLab people as well.

* And they have significant problems of licencing etc.

The Director concludes

- There's not much point in my giving you several nodes if you can't demonstrate that you're making efficient use of one!
- I'm currently set up for very regular problems — yours seem to fit more into a "cloud computing" model than my service's.
- Don't be too depressed — I said that to the MatLab people as well.
* And they have significant problems of licencing etc.
- The 'cloud model' is particularly relevant if you have that 'merge' phase for the results from multiple parallel computations.

The Director concludes

- There's not much point in my giving you several nodes if you can't demonstrate that you're making efficient use of one!
- I'm currently set up for very regular problems — yours seem to fit more into a "cloud computing" model than my service's.
- Don't be too depressed — I said that to the MatLab people as well.
* And they have significant problems of licencing etc.
- The 'cloud model' is particularly relevant if you have that 'merge' phase for the results from multiple parallel computations.

The Director concludes

- There's not much point in my giving you several nodes if you can't demonstrate that you're making efficient use of one!
- I'm currently set up for very regular problems — yours seem to fit more into a "cloud computing" model than my service's.
- Don't be too depressed — I said that to the MatLab people as well.
* And they have significant problems of licencing etc.
- The 'cloud model' is particularly relevant if you have that 'merge' phase for the results from multiple parallel computations.

But I wish you luck!

The algebraist concludes

The algebraist concludes

- I'm somewhat confused about which levels of parallelism we're trying to exploit where.

I'd better talk to my colleagues elsewhere, and try to get together.

The algebraist concludes

- I'm somewhat confused about which levels of parallelism we're trying to exploit where.
- Those numerical people have clearly done much more work in benchmarking than we have.

I'd better talk to my colleagues elsewhere, and try to get together.

The algebraist concludes

- I'm somewhat confused about which levels of parallelism we're trying to exploit where.
- Those numerical people have clearly done much more work in benchmarking than we have.
- I'm also quite impressed by the infrastructure (BLAS etc.) that they have.

I'd better talk to my colleagues elsewhere, and try to get together.