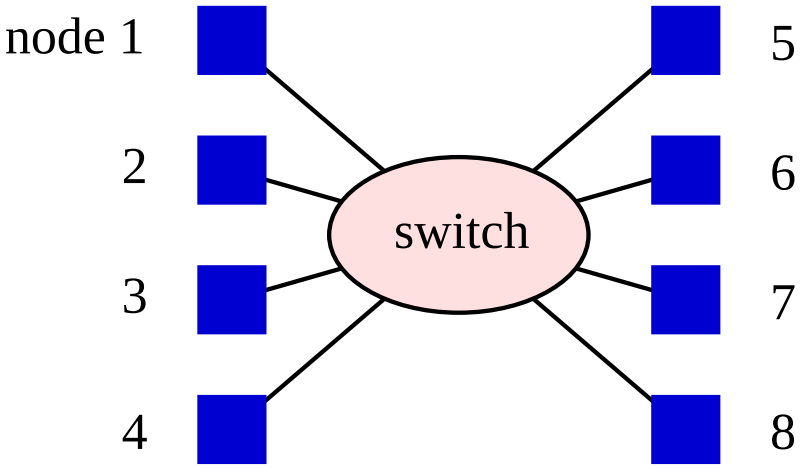
Lattice QCD on PC's?

Martin Lüscher

CERN Theory Division



* One or more PC processors per node
* High-bandwidth network
* Linux operating system with MPI

Lattice 2001, Berlin, 19.–24. August 2001

✓ Cheap processors, memory, ...

✓ Configurable, widely available, proven technology

✓ Standard software environment

but ...

× Floating-point arithmetic is slow

× Memory-to-processor bandwidth limitations

× Network can be a bottleneck

× Engineering problems (heat dissipation, mechanical)

Advanced features of current PC processors

- Vector arithmetic (SSE, 3DNow!, AltiVec)
- On-die cache @ processor clock speed
- Memory-to-cache prefetch
- Streaming memory access

New benchmark results

see also the posters by Di Pierro and Gottlieb

PC with Intel Pentium 4 (1.4 GHz) processor & 256 MB PC800 RDRAM

Using SSE instructions and cache-optimized data layout, L^4 lattice, $L \geq 16$

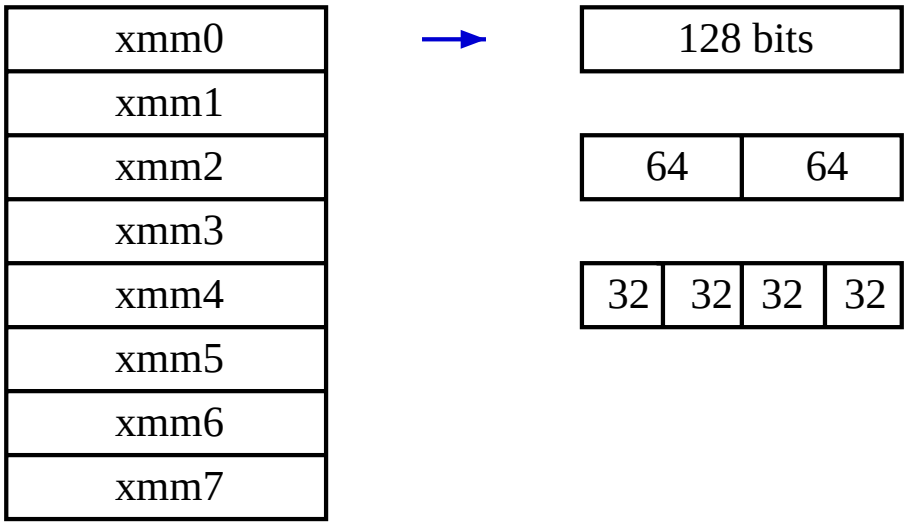
	$(D_{\mathrm{w}} + m_0)\psi$	$\psi_1 + r\psi_2$	$\|\psi\|^2$	CG
32bit	0.93 [1503]	0.13 [363]	0.046 [1042]	2.34 [1292]
64bit	1.71 [814]	0.26 [181]	0.097 [497]	4.40 [687]

(Execution times in μs per lattice point [speed in Mflop/s])

$\sim 10\times$ better than the benchmarks quoted last year (Gottlieb, Lattice 2000)

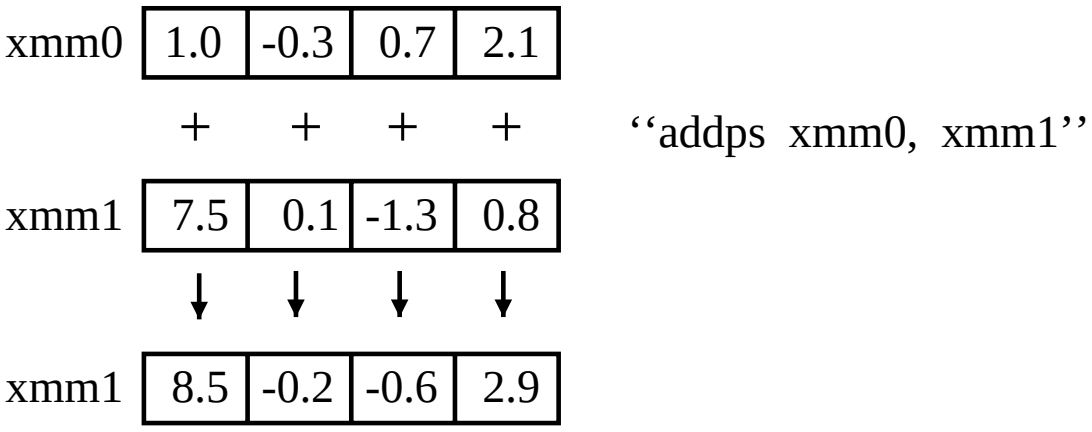
Streaming SIMD Extension (SSE)

- Supported on PIII, P4, Itanium, AMD Sledgehammer
- 8 additional registers for 4 floats or 2 doubles



- SIMD instructions operating on these
- Cache manipulation instructions

SIMD instructions



* Also `subps`, `mulps`, `divps`, `sqrtps`
* IEEE-754 compliant arithmetic
* Instructions for data moving & shuffling

Programming example

Consider the C code

```
for (i=0;i<100;i++)
  a[i]=b[i]+c[i];
```

Using SSE instructions (and GCC) this becomes

```
for (i=0;i<100;i+=4)
  __asm__ __volatile__ (
              "movaps %1, %%xmm0 \n\t"
              "movaps %2, %%xmm1 \n\t"
              "addps %%xmm0, %%xmm1 \n\t"
              "movaps %%xmm1, %0"
              :
              "=m" (a[i])
              :
              "m" (b[i]),
              "m" (c[i]));
```

SU(3) matrix $\times$ vector multiplication

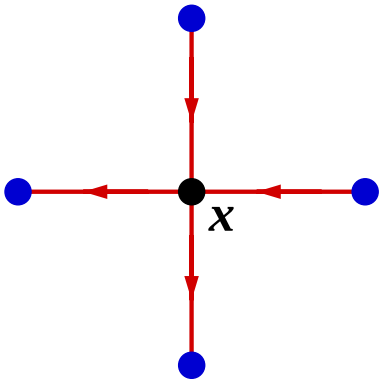
```
#define _sse_su3_multiply(u)
  __asm__ __volatile__ (
          "movss %0, %%xmm3 \n\t"
          "movss %1, %%xmm6 \n\t"
          "movss %2, %%xmm4 \n\t"
          "movss %3, %%xmm7 \n\t"
          "movss %4, %%xmm5 \n\t"
          "shufps $0x0, %%xmm3, %%xmm3 \n\t"
          "shufps $0x0, %%xmm6, %%xmm6 \n\t"
          "shufps $0x0, %%xmm4, %%xmm4 \n\t"
          "mulps %%xmm0, %%xmm3 \n\t"
          "shufps $0x0, %%xmm7, %%xmm7 \n\t"
          "mulps %%xmm1, %%xmm6 \n\t"
          "shufps $0x0, %%xmm5, %%xmm5 \n\t"
          "mulps %%xmm0, %%xmm4 \n\t"
          "addps %%xmm6, %%xmm3 \n\t"
          "mulps %%xmm2, %%xmm7 \n\t"
          "mulps %%xmm0, %%xmm5 \n\t"
          "addps %%xmm7, %%xmm4 \n\t"
           ⋮            ⋮
          "addps %%xmm6, %%xmm4 \n\t"
          "addps %%xmm7, %%xmm5"
          :
          :
          "m" ((u).c11.re),
          "m" ((u).c12.re),
          "m" ((u).c21.re),
          "m" ((u).c23.re),
           ⋮            ⋮
```

On the P4 this code achieves 2.6 Gflop/s [1.8 flop/cycle] !

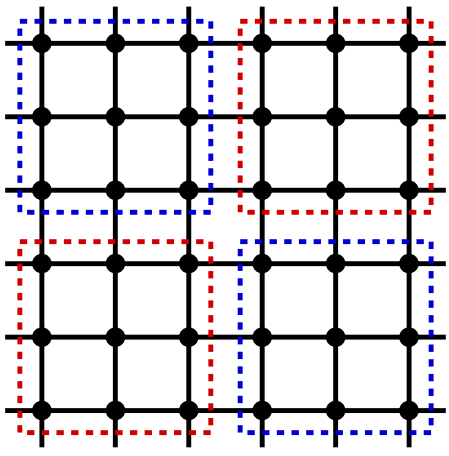
Cache management

Strip-mining

Computation of $(D_{\mathrm{w}} + m_0)\psi(x)$



$\Rightarrow$ data are reused many times



$\Rightarrow$ data blocking enhances the cache-hit probability

Cache management (cont.)

Streaming memory access

Data are fetched from memory cache line by cache line

processor	line size [byte]	# floats	# doubles
Pentium III	32	8	4
AMD Athlon	64	16	8
Pentium 4	128	32	16

⇒ arrange data so that more cache line entries are useful

Memory prefetch

Move data from memory to cache while FPU is busy

```
__asm__ __volatile__ (
        "prefetcht0 %0 :: "m" (*(addr)));
```

⇒ memory latencies can be masked

Memory bandwidth

Benchmark results for streaming memory read

memory type	processor	theoretical [Gbyte/s]	effective [Gbyte/s]
SDRAM PC133	PIII 0.6 GHz	1.1	0.8
DDRAM PC266	Athlon 1.2 GHz	2.1	1.3
RDRAM PC800	P4 1.4 GHz	3.2	2.1

$\Rightarrow$ measured bandwidth $\propto$ processor frequency

Network

Myrinet$^{\mathrm{TM}}$ is currently the preferred choice for the network

- One-way bandwidth ~ 250 Mbyte/s per channel
- Latency around 9 μs
- Switches for up to 128 nodes

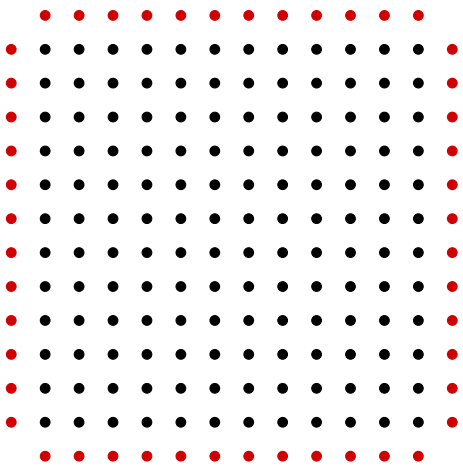
How much bandwidth is needed?

Study case

- 32 nodes, 750 Mflop/s [32bit arithmetic]
- 512 MB local memory
- 200 Mbyte/s bandwidth per channel
- Computation of $(D_{\mathrm{w}} + m_0)\psi$ on a 96×48^3 lattice

Local lattice

8×4 nodes $\Rightarrow$ local lattice is $12 \times 12 \times 48^2$



$$V = 12 \times 12 \times 48^2 \qquad (32\mathrm{MB})$$

$$4 \times 12 \times 48^2 \qquad (11\mathrm{MB})$$

How much bandwidth is needed? (cont.)

Timing

First communication, then computation of $(D_{\mathrm{w}} + m_0)\psi$

$$t_{\mathrm{Comm}} = \frac{96 \text{ byte}}{200 \text{ Mbyte/s}} \times \tfrac{1}{3} V \times 2 = 0.32\,\mu\mathrm{s} \times V$$

$$t_{\mathrm{Comp}} = 2\,\mu\mathrm{s} \times V$$

$\Rightarrow$ communication takes 14% of the total time

Possible improvements

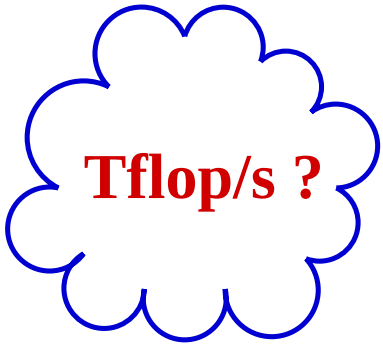
- Try to overlap communication & computation
- Increase network bandwidth

  (max 528 Mbyte/s with current PCI specification)
- Bi-directional communication @ 250 Mbyte/s

More experience & detailed studies are needed!

PC clusters for LQCD *(incomplete)*

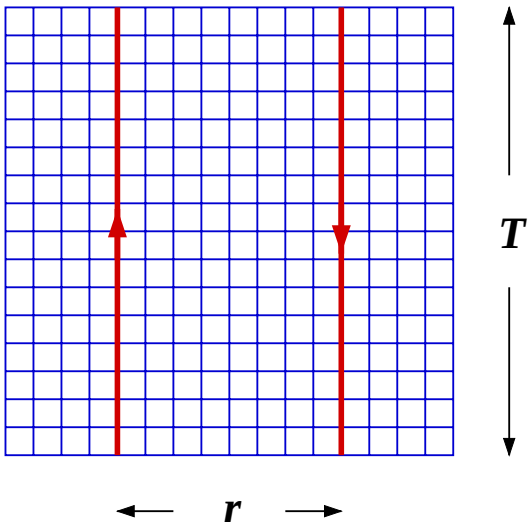
institution	processors	# nodes	status
Fermilab	$2\times$PIII 0.7 GHz	80	running
TC Dublin	$2\times$PIII 1.0 GHz	32	running
MPI Munich	$2\times$PIII 1.0 GHz	8	ordered
DESY-Zeuthen	P4	16	asking for bids
DESY-Hamburg	P4	32	approved
Rome II & CERN	P4	2×64	fast network development
Fermilab	P4	~ 150	approved
Jlab & MIT	$2\times$P4	128	approved



"In general a brute force approach to numerical simulations is not paying and a lot of ingenuity is needed in order to obtain the wanted results"

G. Parisi, Les Houches 1988

Example: Polyakov loop correlation function



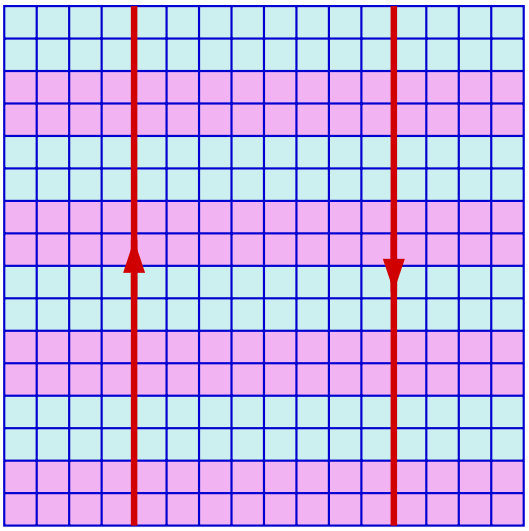
In the confinement phase

$$\langle P(r)^* P(0) \rangle \propto \mathrm{e}^{-\sigma A}, \qquad A = rT, \qquad \sigma \sim 1\mathrm{GeV/fm}$$

$\Delta A = 1\,\mathrm{fm}^2 \quad \Rightarrow \quad$ signal decreases by 6×10^{-3}

$\Rightarrow \quad$ need $\sim 3 \times 10^4$ more statistics

Polyakov loop correlation function (cont.)



"Multilevel" simulation algorithm

$\Rightarrow$ exponential error reduction

M.L. & P. Weisz, hep-lat/0108014

gauge group SU(3)

$T \times L^3$ lattice

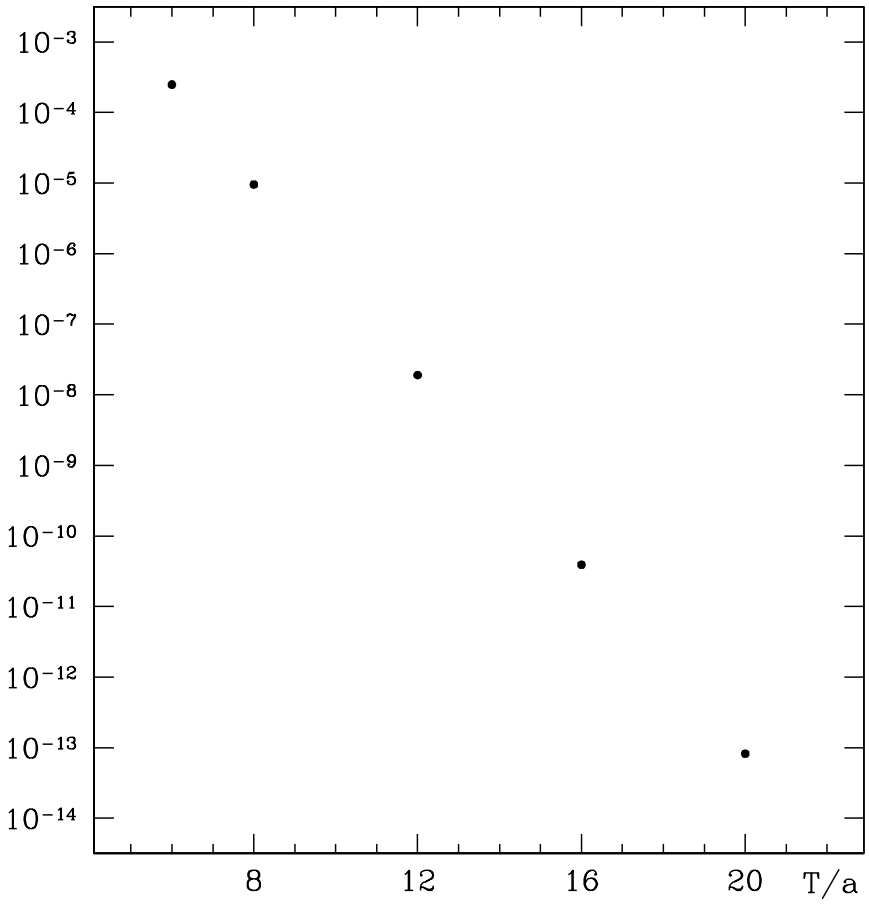
$\beta = 5.7$

$L = 12a$

$r = 6a \simeq 1\,\mathrm{fm}$

statistical errors

are $\sim 2\%$ at all T



Conclusions

- Lattice QCD on PC's? Yes!
- Affordable, versatile, easy to use
- 3–4 \$/Mflop [32bit sustained] including Myrinet$^{\mathrm{TM}}$

Do PC clusters scale to Tflop/s?

Perhaps — need to understand the network issues

However, Tflop/s alone will not solve the hard problems!

$\Rightarrow$ further theoretical & algorithmic innovation is essential