The Technological Frontier in Lattice QCD: Cost Optimized Machines

Thomas Lippert and John W. Negele

*Santorini Workshop on Advanced Computing in Nuclear and Hadronic physics*  
*October 1, 2001*

**Outline**

Cost-Optimized Machines for Lattice QCD

Clusters

  - Motivation
  - Wuppertal / NIC Clusters
  - MIT / Jlab Clusters
  - Future prospects

Custom Parallel Machines

  - QCDSP
  - QCDOC
  - APE mille

Resources and plans
Cost-Optimized Custom Machines for Lattice QCD

- Highly parallel custom machines much cheaper than general purpose supercomputers
  
  regular grid structure

  local communications

  overlapping computation and communications

- Dual approach
  
  optimization of commodity clusters

  fully custom parallel machine

- Robust strategy to pursue both
Motivation for Commodity Based Cluster

- General-purpose architecture and software environment
  - Linux operating system, standard compilers
  - Source code compatibility with workstations
  - Flexibility, ability to implement innovative new approaches
  - Efficient use by everyone in community
  - Ideal for local development by dispersed collaborators
- Double-precision needed for many applications
- Commodity processors and networks offer maximal flexibility
  - Follow best technology in each generation
  - Purchase state-of-art technology any year
    physics or funding motivate it
- Cost-effective:
  - Market forces on processors and communications
  - Processor chip price/performance improving
    like Moore’s Law
  - System integration costs falling
    Large cluster improvement better than Moore’s Law
- Current technology: Alpha → Pentium P4
- Next generation: computer on a chip?
Advanced features of current PC processors

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

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

<table>
<thead>
<tr>
<th>xmm0</th>
<th>xmm1</th>
<th>xmm2</th>
<th>xmm3</th>
<th>xmm4</th>
<th>xmm5</th>
<th>xmm6</th>
<th>xmm7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- SIMD instructions operating on these
- Cache manipulation instructions
SIMD instructions

```
xmm0  1.0  -0.3  0.7  2.1
   +   +   +   +   “addps xmm0, xmm1”
xmm1  7.5  0.1 -1.3  0.8  
   ↓   ↓   ↓   ↓
xmm1  8.5  -0.2 -0.6  2.9
```

* Also subps, mulps, divps, sqrtpps

* IEEE-754 compliant arithmetic

* Instructions for data moving & shuffling
Programming example

Consider the C code

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

Using SSE instructions (and GCC) this becomes

```c
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 × 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 \((D_w + m_0)\psi(x)\)

\[ \Rightarrow \text{data are reused many times} \]

\[ \Rightarrow \text{data blocking enhances the cache-hit probability} \]
## PC clusters for LQCD (incomplete)

<table>
<thead>
<tr>
<th>institution</th>
<th>processors</th>
<th># nodes</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fermilab</td>
<td>2×PIII 0.7 GHz</td>
<td>80</td>
<td>running</td>
</tr>
<tr>
<td>TC Dublin</td>
<td>2×PIII 1.0 GHz</td>
<td>32</td>
<td>running</td>
</tr>
<tr>
<td>MPI Munich</td>
<td>2×PIII 1.0 GHz</td>
<td>8</td>
<td>ordered</td>
</tr>
<tr>
<td>DESY-Zeuthen</td>
<td>P4</td>
<td>16</td>
<td>asking for bids</td>
</tr>
<tr>
<td>DESY-Hamburg</td>
<td>P4</td>
<td>32</td>
<td>approved</td>
</tr>
<tr>
<td>Rome II &amp; CERN</td>
<td>P4</td>
<td>2 × 64</td>
<td>fast network</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>development</td>
</tr>
<tr>
<td>Fermilab</td>
<td>P4</td>
<td>~150</td>
<td>approved</td>
</tr>
<tr>
<td>Jlab &amp; MIT</td>
<td>2×P4</td>
<td>128</td>
<td>approved</td>
</tr>
</tbody>
</table>

Tflop/s ?

M. Lüscher
MIT Prototype Cluster

Joint research project with Compaq

Optimize network and memory costs using 4-processor nodes connected by Myrinet

- $12 \times 4$-processor ES40’s – 64 Gflops
  - EV67 667 MHz processors
  - 8 MB cache/processor
  - 1 GB memory/SMP
  - 9 GB disk
- 500 GB file server
- 4 Alpha 164 workstations
- Myrinet (Lanai 7)
- Installed 12/99–8/00
Jefferson Lab Prototype Clusters

- 16 × 1-processor XP1000’s – 16 Gflops
  - 500 MHz
  - 4 MB cache
  - 256–512 MB memory

- 12 × 2-processor UP2000’s – 32 Gflops
  - 667 MHz
  - 4 MB cache
  - 512 MB memory / SMP

- 400 GB file server
- Myrinet
  - Lanai 9
Custom Machine designed at Columbia

Processor 50 Mflops TI DSP
2MB memory/processor
25 Mbit/sec communications each way to nearest neighbors

Single Precision

Columbia Machine

8192 Nodes
400 GFLOPS Peak
120 GFLOPS Sustained
16 x 16 x 8 x 4
$1.8 M

$15/Sustained MFLOP

Riken Machine

600 GFLOPS Peak
$10/Sustained MFLOP
A block diagram of a single node.

A diagram of the components contained within the gate array chip (NGA).
APE mille

INFN/DESY Successor to APEcento

Processor: 528 Mflops Custom

16-64 MBytes Memory/Proc

Single Precision

44% of Peak (S.P.)

Tower

512 Processors

264 GFlops Peak

8 x 8 x 8
TOPOLOGY

An APEmille machine is a 3D grid of processing nodes with hardware data links between the 3D first neighbouring nodes. The smallest machine is the single Processing Board (PB) where are placed 8 processing nodes with a 2x2x2 topology, (cube). Arranging togheter more PBs it's possible to have more complicated, greater and faster machines.

- Hierarchical Hardware Topology
- Performances

• Hierarchical Hardware Topology

The yellow cubes in this rappresentation are single APEmille boards with their 8 processing nodes supposed as placed at the vertices of the cube.

- Board: 8 Nodes (2*2*2 topology)

- SubCrate: 4 Boards (2*2*8 topology)

- Crate: 4 SubCrates - 16 Boards (2*8*8 topology) 66 GFlops

- Tower: 4 Crates - 64 Boards (8*8*8) 264 GFlops
- APEMille: 4 Towers - 256 Boards (32*8*8)

• Performances

The following table reports the performance of the different configuration of an APEmille system: the maximum performance are about 1TeraFlops for the 66 MHz machine, and could be 1.6TFlops in a future 100 MHz release.
PROCESSING BOARD

- Scheme of a processing board

For each APEmille PB a CPU takes care of the program flow and drives the communication protocol with the host network. The CPU broadcasts a VLIW (Very Long Instruction Word) to the 8 Processing Nodes located on each PB. At the same time, as a result of its calculations, the CPU broadcasts a Global Address to the Processing Nodes. The Nodes can access their own local memory using this Global Address either directly or after adding a local offset to it, thus producing a local address.

The 8 Processing Nodes of a Board are arranged in a cubic lattice (a 2x2x2 topology). Each processing node has an arithmetic and logic unit including floating point adders, multipliers, a large multiport register file plus a memory controller with address generation capability, interfacing to local memory. Integer and bitwise operations and local addressing are some of the new main features of the Processing Nodes.

The first APEmille systems, based on presently available Synchronous Dynamic Ram and VLSI ASIC technologies, will adopt a 66 MHz, 528 MFlops processor. We expect however that the clock frequency may eventually increase up to 100 MHz. A 2048 100 MHz nodes system would have a peak performance of 1.6 TeraFlops.

According to our application requirements, the size of the local memory of each Processing Node will range from 2 to 8 MWord. Through a Communication Device each Processing Node directly accesses the data on its own 6 neighbouring Nodes. Some routing capabilities allow of the communication device (slower) access to far-away nodes.

The Processing Board hosts:

- 1 CPU
- 8 Processing Nodes,
- 1 Communication Device
- required memories.

- Scheme of connections among the 8 nodes of a PB
**Cmille** is the custom chip which takes care of the remote data communications between processing nodes.

**Tmille** is the custom processor which controls the instruction flow and global addressing of each APEMille SIMD partition (a minimum of 8 processing nodes). Tmille is connected to the Host, to its own data memory, to an instruction memory and to the Communication Device. It also drives the Address Bus. There is a major difference from the Ape100 machine where one controller CPU normally drives 128 nodes and is housed in a different board.

The **Processing Node** is composed of a floating point processor (called JMILLE) which drives its own Local Memory, receives Instructions and Global Addresses from TMILLE and communicates with the Communication Device (CMILLE).

The Communication Device also has a channel connected to the CPU, and through this pathway, to the Host. Through this pathway the Host loads the program and data memories of the floating point processor.

All the CPUs and all the Nodes of a SIMD machine partition execute the same instruction. Each PB sends its Local Signals to a Root Board and receives Global Signals from it, in order to manage Synchronisation, Exceptions and Global Flow Control Conditions.

Each Processing Board interfaces the Host through a dedicated synchronous channel (APE Channel) going from the CPU to the Host Interface.
PROCESSING NODE

Each Processing Node is composed of a JMILLE floating point processor, attached to a SDRAM local memory. The Memory Controller inside Jmille generates addresses for the External Memory summing up a Global Address given by Tmille and a Local Address computed by Jmille itself. This way each APEMille Processing Node is able to generate a different Memory Address. The five addresses needed by the Multiport Register File are fixed by our compiler at compile time, and therefore distributed to Jmille inside the Program Word. This is obtained by means of our VLIW (Very Long Instruction Word) compiling technology.

Each Processing Node generates Status Signals (Global conditions, Exceptions etc.). These Status Signals are collected into a Global Status Return managed by the Control System. This connection allows the Control System to execute flow control instructions based on simultaneous logical conditions produced by the set of Processing Nodes. The instruction word read by JMILLE specifies the set of addressed location inside the large multiport Register File internal to JMILLE, and controls the Arithmetic Devices inside the processing node. Moreover it specifies Local Conditional Operation, Local Addressing functions, and Special Arithmetic function calculations to be performed by JMILLE.

- **Scheme of a Processing Node**

- **Jmille**: is the APEMille custom arithmetic processor with hardware supporting arithmetic, logical and bitwise operations on complex, double precision, single precision and integer data types.

- **External Memory (ETM)**: A SDRAM memory, directly attached to Jmille and controlled by Jmille itself, where all the data related to the node processing are stored.
A short list of some large APEmille existing installations and of plans for the near future.

**END OF 2001 (Gflops Peak)**

<table>
<thead>
<tr>
<th>Site</th>
<th>peak performance</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rome</td>
<td>260 Gflops</td>
<td>running</td>
</tr>
<tr>
<td>Zeuthen</td>
<td>130 Gflops</td>
<td>running</td>
</tr>
<tr>
<td>Rome II</td>
<td>65 Gflops</td>
<td>running</td>
</tr>
<tr>
<td>Bielefeld</td>
<td>80 Gflops</td>
<td>running</td>
</tr>
<tr>
<td>Milano/Parma</td>
<td>130 GFlops</td>
<td>planned, Dec. 2000</td>
</tr>
<tr>
<td>Pisa</td>
<td>130 Gflops</td>
<td>planned, Dec. 2000</td>
</tr>
<tr>
<td>Rome II</td>
<td>+ 65 Gflops</td>
<td>planned, Dec. 2000</td>
</tr>
<tr>
<td>Zeuthen</td>
<td>+ 260 Gflops</td>
<td>planned, Spring 2001</td>
</tr>
</tbody>
</table>

1380

2042
QCDOC (QCD On a Chip)

- Natural extension of custom QCDSP machines at Columbia and RIKEN

- Partnership with IBM since December 1999 to design and manufacture ASIC
  
  Processor: 1 Gflops peak 440 Power PC
  4 MB on-chip memory
  8 Gbits/sec interprocessor communication

- Schedule
  
  ASIC December 2001
  Prototype March 2002
  10 Tflops sustained QCDOC at BNL October 2003
  Hardware cost: $10 M

- Collaboration
  
  Columbia University
  RIKEN
  IBM
  Edinburgh
  Fermilab
QCDOC ASIC DESIGN

- 4 MBytes of Embedded DRAM
- 8 Gbyte/sec Memory/Processor Bandwidth
- 1 Gflops Double Precision RISC Processor
- 2.6 GByte/sec Interface to External Memory

2.6 GByte/sec EDRAM/SDRAM DMA

24 Link DMA Communication Control

24 Off-Node Links 12 Gbit/sec Bandwidth

Bootable Ethernet Interface

100 Mbit/sec Fast Ethernet

Complete Processor Node for QCD Supercomputer on a Single Chip Fabricated by IBM

Mission-critical, custom logic (hatched) for high-performance memory access and fast, low-latency off-node communications is combined with standards-based, highly integrated commercial library components.
Lattice resources available internationally


State-of-art quenched calculation 50 Gflops-yrs
Largest NERSC FY00 QCD allocation 2 Gflops-yrs
Largest NSF FY00 QCD allocation 10 Gflops-yrs

<table>
<thead>
<tr>
<th>Country</th>
<th>FY00 Sustained Gflops</th>
</tr>
</thead>
<tbody>
<tr>
<td>Germany</td>
<td>264</td>
</tr>
<tr>
<td>Italy</td>
<td>554</td>
</tr>
<tr>
<td>Japan</td>
<td>980</td>
</tr>
<tr>
<td>U.K.</td>
<td>41</td>
</tr>
<tr>
<td>U.S. (Columbia)</td>
<td>120</td>
</tr>
<tr>
<td>U.S. (RIKEN)</td>
<td>180</td>
</tr>
<tr>
<td>U.S. (open)</td>
<td>20</td>
</tr>
</tbody>
</table>

European Committee for Future Accelerators (ECFA) report

- Several 10 Tflops computers in FY03
- UKQCD committed to buy QCDOC
<table>
<thead>
<tr>
<th></th>
<th>QC DSP</th>
<th>QC DOOC</th>
<th>APEmille</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Processor</strong></td>
<td>TI DSP</td>
<td>Power PC 440</td>
<td>Custom</td>
</tr>
<tr>
<td></td>
<td>50 MF</td>
<td>1 GF</td>
<td>528 MF</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>2 MB/proc</td>
<td>4 MB/proc</td>
<td>16-64 MB/proc</td>
</tr>
<tr>
<td><strong>Interconnect</strong></td>
<td>25 Mbit/sec x16</td>
<td>500 Mbit/s x24</td>
<td></td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>30%</td>
<td>50% goal</td>
<td>44%</td>
</tr>
<tr>
<td><strong>Machines</strong></td>
<td>400 GF</td>
<td>plan 5 TF</td>
<td>264 GF</td>
</tr>
<tr>
<td></td>
<td>600 GF</td>
<td>10 TF</td>
<td></td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>$10-15/sust. MF</td>
<td>$1/sust. MF</td>
<td></td>
</tr>
</tbody>
</table>
### 4 Compute Engines for Lattice-QCD

<table>
<thead>
<tr>
<th>Engine</th>
<th>Description</th>
<th>Year(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>APE100:</td>
<td>INFN, 25.6 Gflops (QH4 largest unit) (1994)</td>
<td></td>
</tr>
<tr>
<td>CP-PACS:</td>
<td>Tsukuba Center for Computational Physics and HITACHI, 600 Gflops (1996)</td>
<td></td>
</tr>
<tr>
<td>Columbia-QCDSP:</td>
<td>Columbia University, 100 Gflops (largest unit?) (1997)</td>
<td></td>
</tr>
<tr>
<td>Cluster Computers:</td>
<td>e.g. ALiCE, 160 Gflops (2000)</td>
<td></td>
</tr>
<tr>
<td>APEmille:</td>
<td>INFN/DESY, 64 Gflops (largest unit) (2001)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Engine</th>
<th>Description</th>
<th>Year(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>QCDDOC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Columbia-QCDDSP:</td>
<td>Columbia University/UKQCD, 10 Tflops (2003)</td>
<td></td>
</tr>
<tr>
<td>apeNEXT:</td>
<td>INFN/DESY, 5 Tflops (2004)</td>
<td></td>
</tr>
<tr>
<td>Clusters:</td>
<td>Jefferson Lab/MIT, FNAL, 10 Tflops (2004/5)</td>
<td></td>
</tr>
</tbody>
</table>