Computer Physics Communications 147 (2002) 402–409

Computer Physics Communications

www.elsevier.com/locate/cpc

APE computers—past, present and future

F. Bodin[a], Ph. Boucaud[b], N. Cabibbo[c], G. Cascino[d], F. Calvayrac[e], M. Della Morte[f], A. Del Re[d], R. De Pietri[g], P. Deriso[h], F. Di Carlo[c], F. Di Renzo[g], W. Errico[i], R. Frezzotti[f], U. Gensch[k], T. Giorgino[k], M. Guagnelli[h], N. Hervé[a], K. Jansen[j,*], H. Kaldass[k,1], A. Lonardo[c], M. Lukyanov[k,2], G. Magazzú[i], J. Micheli[b], V. Morénas[l], L. Mori[g], F. Palombi[h], N. Paschedag[k], J. Pech[m], O. Péne[b], R. Petronzio[h], D. Pleiter[j], F. Rapuano[c], D. Rossetti[c], C. Roiesnel[n], L. Sartori[i], H. Simma[k], F. Schifano[i], R. Tripiccione[o,*], P. Vicini[c], W. Wegner[k]

[a] *IRISA/INRIA, Campus Univ. de Beaulieu, Rennes, France*
[b] *LPT, University of Paris Sud, Orsay, France*
[c] *INFN, Sezione di Roma, Italy*
[d] *Dipartimento di Ingegneria elettronica, University of Roma "Tor Vergata", Italy*
[e] *LPEC, Univ. du Maine, Le Mans, France*
[f] *Physics Department, University of Milano Bicocca and INFN, sezione di Milano, Italy*
[g] *Physics Department, University of Parma and INFN, gruppo collegato di Parma, Italy*
[h] *Physics Department, University of Roma "Tor Vergata" and INFN, Sezione di Roma II, Italy*
[i] *INFN, Sezione di Pisa, Italy*
[j] *NIC/DESY Zeuthen, Germany*
[k] *DESY Zeuthen, Germany*
[l] *LPC, Univ. Blaise Pascal and IN2P3, Clermont, France*
[m] *CERN, Geneva, Switzerland*
[n] *CPT, École Polytechnique, Palaiseau, France*
[o] *Physics Department, University of Ferrara and INFN, Sezione di Ferrara, Italy*

Abstract

APE is a family of supercomputers architecturally optimized for the numerical simulation of quantum field theories. Current generation APE systems (APEmille) have been commissioned at several European sites. When all planned systems are installed, later this year, a total peak processing power of about 2 TFlops will be available. A new generation system, apeNEXT, is under development. It adds several new features to the established APE architecture. Performance will be boosted towards the 10 Tflops range. © 2002 Published by Elsevier Science B.V.

Keywords: Parallel computing; Lattice gauge theories

[*] Corresponding authors.
E-mail addresses: karl.jansen@desy.de (K. Jansen), tripiccione@pi.infn.it (R. Tripiccione).
[1] On leave of absence from University of Texas, Austin.
[2] On leave of absence from JINR, Dubna.

0010-4655/02/$ – see front matter © 2002 Published by Elsevier Science B.V.
PII: S0010-4655(02)00314-4

1. Introduction

APE is one of several projects (for a review see [1]) in the theoretical physics community that have developed massively parallel high-performance computers essentially from scratch. The driving force why physicists develop and build computers by themselves is the success of numerical simulations in understanding the interactions of elementary particles, in particular their strong interactions described by quantum chromodynamics (QCD).

In the absence of closed-form analytical solutions for theories, like QCD, one of the most interesting approximation schemes is a re-formulation of the theory on a discrete lattice (see [2] for a short introduction, or [3]). The original theory is recovered as the lattice spacing a goes to zero. This approach, pioneered by K. Wilson more than 25 years ago [4], is the starting point for Lattice Gauge Theory (LGT). This discrete and computer-friendly formulation of quantum field theory has triggered an immense activity. (See [5] for an overview.)

The phenomenona investigated with such simulations range from the permanent confinement of quarks inside hadrons to the cosmological phase transition that occurred in the early phases of the universe or in matter under extreme conditions as produced in heavy-ion collision experiments. Within the framework of LGT, fundamental parameters of QCD, like the masses of quarks or the strength of the running strong coupling α_s (see Fig. 1), have been computed from first principles. Also, theoretical concepts such as spontaneous chiral symmetry breaking and even the mathematical structure of the theory itself can be tested with modern simulation techniques.

One of the big challenges is the determination of weak matrix elements of hadronic states to understand the interplay between weak and strong interactions. Problems like the $\Delta I = 1/2$ rule or the violation of CP symmetry are still open. The study of the heavy quark semileptonic decays is crucial for the determination of the Cabibbo–Kobayashi–Maskawa angles which are basic parameters of the Standard Model. A further example is the non-leptonic decay $K \to \pi\pi$, relevant to understand CP violation [7]. Experiments show evidence of direct CP violation in kaon decays through the measurement of a non-vanishing value of the ϵ'/ϵ ratio. Whether this result can be accounted for in the Standard Model is not yet clear.

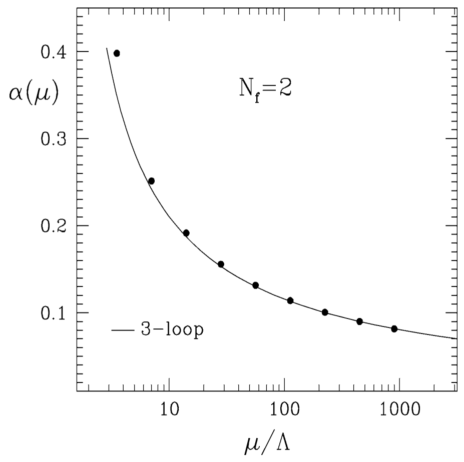

Fig. 1. The strength of the coupling as a function of the energy at which the system is probed. From Ref. [6]

Many of the current LGT projects focus on the simulation of QCD with dynamical fermions. Because of the limits in available computing power one is forced in many cases to apply the so-called quenched approximation, where the effects of vacuum fermion loops are neglected. Although the currently available computing resources allow to relieve this approximation, it will be extremely hard to lower the masses of the dynamical quarks towards their physical values. It will be even more difficult to reduce the lattice spacing and to do simulations closer to the continuum limit. A tremendous amount of computer power is required to overcome these limitations. A panel of the European Committee for Future Accelerator (ECFA), which proposed an ambitious research program for the coming years, estimates that European research groups would need about 10 Tflops of compute power [8].

In order to make these computing resources available at a reasonable price, various research groups have engaged in the development of supercomputers which are specifically optimized for their applications.

In this paper we describe the Array Processor Experiment (APE) project, which was started in the mid eighties by the Istituto Nazionale di Fisica Nucleare (INFN) and is now carried out within the framework of a European collaboration with DESY and the University of Paris Sud.

APE100, the second generation of APE supercomputers, has been the leading workhorse of the European lattice community since the middle of the 1990s. Commissioning of several large European installations of the following generation of APE computers, APEmille, is almost completed, making another 2 Tflops of computing power available to the LGT community.

In order to keep up with future and growing requirements, the development of a new generation of a multi-TFlops computer for LGT, apeNEXT, has started.

In the remainder of this paper, we first describe in general terms the key features of a computer architecture optimized for simulations in LGT, and then highlight the APEmille implementation. We then briefly cover our experience in operating APEmille. A further section describes apeNEXT, followed by a discussion of some software issues. The final section contains our concluding remarks.

2. LGT computer architectures

The bulk of the numerical work to simulate LGT goes into the solution of a set of linear equations

$$M\chi = \phi \tag{1}$$

for a number of right-hand sides ϕ. The complex valued matrix M (usually referred to as the Dirac operator) is very large ($12L^4 \times 12L^4$, where L is the number of lattice points used to discretize the physical space), but very sparse. Nowadays simulations use $16 \lesssim L \lesssim 64$. A convenient way to label an entry of M is $M^{(x,y,z,t),i}_{(x',y',z',t'),i'}$, where i (i') are internal indices, while the set (x, y, z, t) labels a physical point in the four-dimensional lattice. Typically, the only non-zero elements of M are those corresponding to nearest-neighbor points in the four-dimensional lattice. The solution to (1) can be found using a variety of iterative algorithms (Krylov space solvers), such as minimal residue or conjugate gradient. The sparseness of the matrix makes parallelization of these algorithms straightforward.

The algorithmic structure outlined above can be used as a guideline to shape an optimal architecture for an LGT number-cruncher:

– Fine-grained parallelism can be extended to an extremely high number of nodes. Regular meshes of processors of dimensionality up to 4 allow an immediate geometric mapping of the physical variables. The only heavily used pattern of data communication between processing nodes is among nearest-neighbor processors in the mesh.
– Most of the time all nodes in the mesh execute the same sequence of operations. Simple control strategies like SPMD (Single Program Multiple Data) or even SIMD (Single Instruction Multiple Data) are therefore appropriate. Note that, once the physical variables are allocated evenly on the processing nodes, load balancing (on average) is automatically achieved on the system.
– Each node in the mesh must be heavily biased towards floating-point performance. Fat arithmetic operators (performing more than one arithmetic operation per clock cycle) are an effective trick since almost all arithmetics involve complex numbers.
– Kernel programs sweep across huge data sets at each iteration, so data caches do usually not have a very high hit rate. However, critical computations apply several arithmetic operators repeatedly to a relatively small set of data words (for instance, the innermost loop in the calculation of the Dirac operator processes 360 real data words with a total of about 2400 floating-point operations). The data access pattern is orderly and predictable, so memory prefetch mechanisms can be efficiently used.
– Simple styles of algorithm coding foresee rather frequent access to small blocks of remote data, so low latency in the data exchange between neighboring nodes is an important feature.

In brief, a good LGT compute engine can be implemented as a massively parallel system and its processing elements require only a subset of the functionalities encountered in standard off-the-shelf processors. For these reasons, the development of scalable LGT systems based on custom processors, that can be easily assembled by hundreds or thousands, has clearly been a winning choice in the last decade.

In the next section, we describe the details of APEmille, the present generation LGT machine of the APE project.

3. Features of the APEmille architecture

APEmille is a massively parallel computer which is primarily optimized for simulating QCD. The architecture is SIMD and all nodes run strictly synchronously at a moderate clock frequency of 66 MHz. The communication network has a three-dimensional topology and offers a bandwidth of 66 MBytes/s/node. The smallest APEmille unit is a processing board with $2 \times 2 \times 2$ nodes (see Fig. 2). The largest stand-alone systems built until now consist of $4 \times 8 \times 8$ nodes (see Fig. 3). Three different integrated circuits (ASICs) have been custom-developed for APEmille. Program execution is controlled by a control processor, which also performs the subset of integer arithmetics common to the whole SIMD partition. Computations using local integers and all floating-point operations are done in parallel by all computing nodes. At each clock cycle, the arithmetic processors are able to complete the operation $a \times b + c$, where a, b, c are single precision (32 bit) complex operands. This gives a peak performance of 528 Mflops per node. Each node has 32 MBytes of local memory and a very large register file, holding up to 512 data values. Remote communications between the nodes are implemented as direct memory access which is controlled and routed by the communication processors.

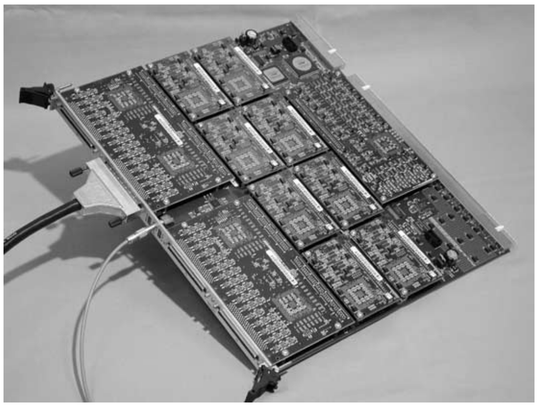
Fig. 2. APEmille board with 8 processing nodes, 2 communication modules, 1 control processor and a PCI interface.

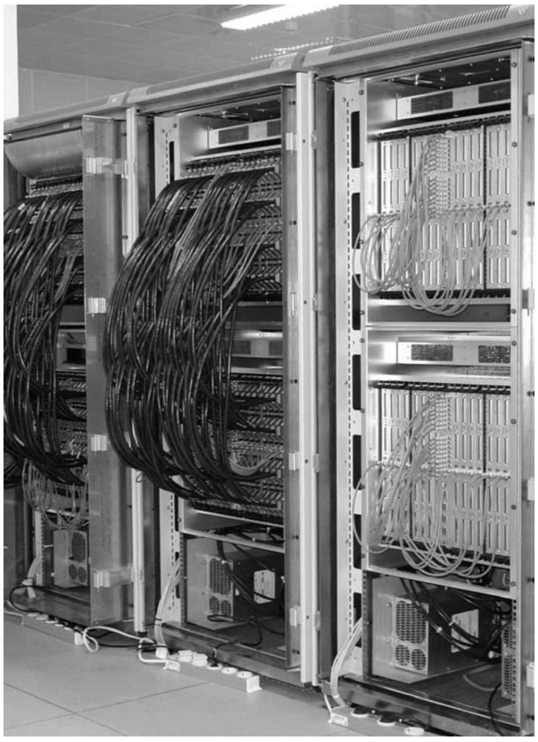
Fig. 3. Three APEmille racks each with 256 nodes and 130 Gflops peak performance.

The processors are controlled by very long instruction words (VLIW). This allows efficient scheduling of the micro-code at compile time. Much effort has therefore been put into the development of software tools for generating efficient code (see later for more details). Loading of the executables and all other operating system services are handled via PCs running Linux. One host PC per four boards is directly attached to the APEmille backplane. It uses a PCI bus to communicate with the processing boards. The system services are controlled by one master PC per machine.

4. APEmille installations

APEmille systems have been installed at several sites as detailed in Table 1. Additional machines will be commissioned in fall 2001. The total available processing power will be of the order of 2 Tflops.

Most APEmille machines consist of stand-alone machines with 256 ($4 \times 8 \times 8$) nodes and 8 host PCs. The operating system also allows them to be partitioned into independent subsystems with 128 ($2 \times 8 \times 8$), or 8 ($2 \times 2 \times 2$) nodes. In most cases, simulation results are stored on file servers accessed by local area networks. A typical file server has

Table 1
APEmille installations. For each site, we list the total installed peak processing power available now and planned for end 2001. Sites in bold do not belong to the APE collaboration

Site	Gflops installed	Gflops planned end 2001
Rome I	455	650
DESY Zeuthen	455	550
Pisa	130	260
Rome II	130	260
Milano	65	130
Bari	65	65
Univ. Paris Sud	16	16
Bielefeld	80	140
Swansea	65	65
Grand total	1461	2136

1–2 Tbytes of data disks on line. Power consumption is very low (less than 30 W/Gflops) and the footprint of a 130 Gflops machine is about 0.7 m^2. For these reasons, APEmille machines are simply air cooled and do not need complex infrastructure.

Normally up-times of 1–2 months or more can be achieved. Hardware maintenance is typically limited to simple replacement of ageing modules and is therefore cheap, both in terms of hardware costs and manpower.

5. apeNEXT

The next-generation APE system, apeNEXT, is currently in the development phase. The basic architectural features are essentially unchanged, while technology improvements boost performance substantially: each node will have a peak performance of 1.6 Gflops in 64-bit double precision, while the communication bandwidth between neighboring nodes is 200 Mbyte/s. We envisage large apeNEXT systems with 2000 processing nodes, delivering a peak performance of 3.2 Tflops.

The apeNEXT architecture is similar to APEmille, except for two key features.

First, apeNEXT is a SPMD (as opposed to SIMD) system. Each processing node is a fully independent processor, with a full-fledged flow-control unit (and of course a number-crunching unit). The node has access to its own memory bank, where both program and data are stored. It executes its own copy of the program at its own pace. Nodes are synchronized only when a data-exchange operation is performed. This architecture may be labeled as a distributed-memory parallel processor, in which nodes exchange data through some sort of “message-passing” scheme. Global program consistency dictates that each program section sending data to a remote node is matched by corresponding instructions on the destination node, that explicitly receives the data packet. However, the latency associated to a “message” is extremely short, of the order of 2 to 3 times the latency associated to an access to local memory. For this reason, the actual data rate between nodes is bandwidth-limited (as opposed to latency-limited) even for short packets, so sequences of short accesses can be freely programmed without significant performance losses.

The second main improvement in the architecture is the possibility of routing all read memory accesses (to local or remote nodes) through a receiving queue, which can be later accessed by the processor with zero latency. This feature is mainly used to perform data prefetch in critical kernel loops. The basic idea here is that all data items needed to perform iteration $(i+1)$ of the loop are prefetched during iteration i and stored into the queue. When iteration $(i+1)$ is performed data will be immediately available to the processor, effectively hiding any latency effect.

Note that some of the memory accesses will be local, and some will be remote. They are started in sequence, but they may complete in a different order (a remote access may take longer than a local one). However, the queue mechanism automatically ensures that data are delivered to the processor in the same order in which they were requested from (remote or local) memory.

The overall structure of an apeNEXT system is similar to APEmille. We have a three-dimensional array of processors, with data links connecting nearest-neighbor nodes in all directions. Periodic boundary conditions are applied.

The complete processing element is built in just one custom-designed integrated circuit, called J&T, connected to a memory bank based on 256 Mbit Double Data Rate (DDR) Dynamic Ram chips. The planned size of the memory bank belonging to each node is 512 Mbytes.

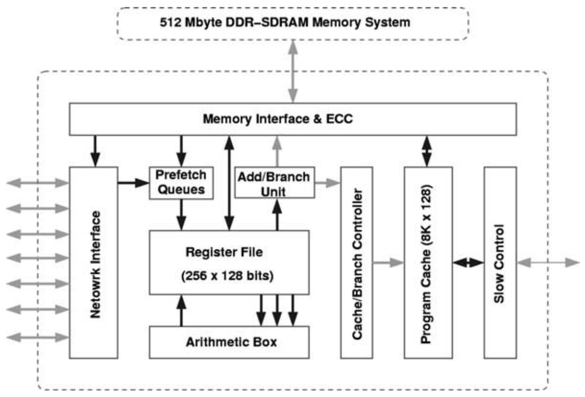

Fig. 4. Block diagram of J&T, the single-chip processor used in apeNEXT.

The block diagram of the J&T chip is shown in Fig. 4. The main elements of the node are:

– A large register file (256 registers containing pairs of 64 bit words). All operands for the arithmetic unit arrive from the register file and all results are written back here.
– An arithmetic box, that computes the "normal" floating-point operation $a \times b + c$. The operands are either complex values, or pairs of real values. In the former case, the normal operation is equivalent to 8 standard floating-point operations. In all cases floating-point data follow the IEEE standard for double-precision (64 bit) values.
– A program cache, where code belonging to critical kernels can be kept in the processor to save precious memory bandwidth for data access.
– A network interface, that handles 7 data links. Each link is bi-directional and the bandwidth is 200 Mbyte/s per each direction. Six links are used to connect the processing node to its neighbor, while the seventh link is used for input-output to the host system.
– The prefetch queues, described earlier in the text.
– A slow serial interface, based on the I2C standard, used for system initialization, debugging and exception handling.

A set of 16 processing nodes, assembled in a $4 \times 2 \times 2$ geometry, makes up the basic hardware building block for the system, known as the processing board (PB). Each PB has an access point to a data link (the seventh link mentioned above). These data links can be connected to standard PCs, using a PCI interface. A real system may have one or more such connections, depending on the I/O bandwidth required by the specific application. All PBs also have an I2C interface for system boot, debugging and exception handling. This interface is also connected to a PC in the hosting cluster.

Blocks of 16 PBs are housed inside a "system crate". Such a system is globally a 256 node, 400 Gflops apeNEXT machine. Finally, several crates can be connected together to form larger systems.

apeNEXT development is well advanced. All hardware components have been completely designed. Prototypes of the J&T processor are expected in spring next year, and a large system prototype is forecast for late 2002.

6. Software

All APE machines, since the mid-eighties, have been traditionally programmed in the high-level TAO language. This language has a basic syntax very similar to the C language and provides all features needed to write efficient LGT simulation programs, as well as some useful extensions. Most useful is the ability to enlarge the language syntax in an object-oriented style, which allows the definition of complex data types and mathematical operators (say, the product of a matrix and a vector).

Parallelism (in SIMD/SPMD form) is automatically enforced in the language by the convention that each variable declared by the code is instantiated in the memory space of each node. All processors execute the same flow of instructions, operating on their private data.

TAO has been heavily used by the LGT community and a large set of simulation and analysis programs are available. For this reason, TAO will also be made available for apeNEXT.

We also plan to develop a more traditional C compiler. The key advantage is of course easier portability of new programs to the APE systems. We also want to make program migration between APE and traditional computer systems simple and easy (see later for further comments on this point). A C compiler was not developed on previous generation APE machines,

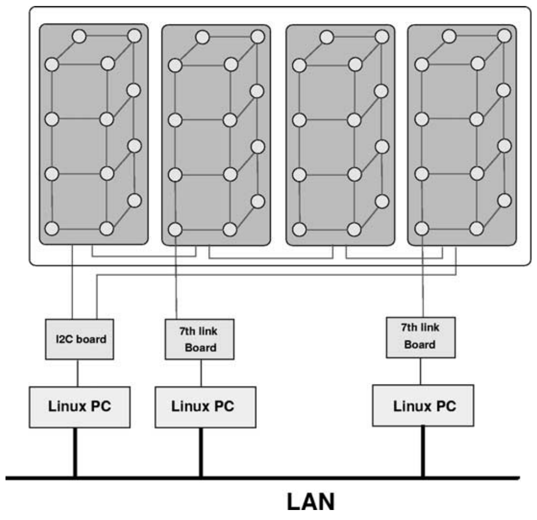

Fig. 5. Block diagram of a complete apeNEXT system, including the fast I/O and slow control channels to the host system, based on a cluster of PCs.

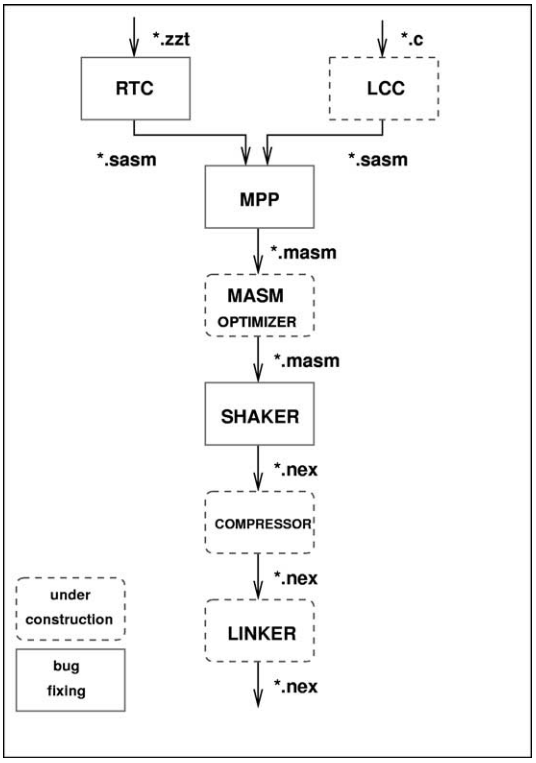

Fig. 6. The main elements of the compilation chain for apeNEXT. The APE-specific TAO language will be supported, as well as a subset of C.

since architectural details made the effort almost hopeless.

We are planning a common compilation chain, for both TAO programs and C programs, as shown in Fig. 6. For TAO, we use the same front-end (RTC) as in APEmille, which is well-tested and rather efficient in exploiting memory accesses in burst mode for contiguous data structures. The front-end module of the C compiler will be derived from well-known open-source compilers, like *lcc* or GNU-*gcc*. After pre-processing and expansion by MPP, the assembly code undergoes two levels of optimization. First, high-level optimization tricks, like loop unfolding, merging of arithmetic operations into "normal form", register-move removal, etc. are performed on the assembly code. The assembly code is then scheduled in order to make optimal use of the machine pipelines. This operation (known as the "shaking" of the code) is particularly efficient thanks to the VLIW (Very Long Instruction Word) control structure of the processor. The code is then compressed and, if needed, linked.

Our goal is to have both compilers on almost equal footing, as far as sustained efficiency is concerned.

7. Outlook

APEmille provides substantial computing power for the LGT community in Europe. These systems are remarkably efficient and cheap as a result of an architecture closely and accurately tailored to the specific requirements of the LGT simulation algorithms. This approach is followed in the development of a future generation system, based on the traditional APE architecture, which will meet the computing requirements of the LGT community in the next 3–4 years.

In the long run, it is not clear whether dedicated LGT architectures will retain their advantages, given the dramatic performance improvements of commercial micro-processors in recent years, especially if all their hardware features are fully exploited [9]. Intermediate size clusters of PCs can be easily assembled today. The real challenge here is the development of a communication fabric able to handle thousands of interconnected PC's. Part of the APE collaboration is working along these lines.

References

[1] N.H. Christ, Computers for lattice QCD, Nucl. Phys. (Proc. Suppl.) 83–84 (2000) 111.
[2] R. Gupta, General physics motivations for numerical simulations of quantum field theory, Parallel Comput. 25 (1999) 1199.
[3] I. Montvay, G. Münster, Quantum Fields on a Lattice, Cambridge University Press, 1994.
[4] K.G. Wilson, Confinement of quarks, Phys. Rev. D 10 (1974) 2445.
[5] T. Bhattacharya et al., Contents of Lattice 2000 Proceedings, Nucl. Phys. B (Proc. Suppl.) 94 (2001); hep-lat/0104009.
[6] A. Bode et al., ALPHA Collaboration, First results on the running coupling in QCD with two massless flavours, hep-lat/0105003.
[7] M. Papinutto, Talk at Lattice 2001, Nucl. Phys. B (Proc. Suppl.), to be published.
[8] F. Jegerlehner et al., Requirements for high performance computing for lattice QCD: Report of the ECFA working penal, Preprint ECFA/99/200.
[9] M. Lüscher, Lattice QCD on PCs?, Talk at Lattice 2001, Nucl. Phys. B (Proc. Suppl.), to be published.