Our discussion of parallel computer architectures starts with the recognition that parallelism at different

levels can be exploited. These levels are:

• Bit-level parallelism. The number of bits processed per clock cycle, often called a word size, has

increased gradually from 4-bit processors to 8-bit, 16-bit, 32-bit, and, since 2004, 64-bit. This has

reduced the number of instructions required to process larger operands and allowed a significant

performance improvement. During this evolutionary process the number of address bits has also

increased, allowing instructions to reference a larger address space.

• Instruction-level parallelism. Today’s computers use multi-stage processing pipelines to speed

up execution. Once an n-stage pipeline is full, an instruction is completed at every clock cycle.

A “classic” pipeline of a Reduced Instruction Set Computing (RISC) architecture consists of five

stages2: instruction fetch, instruction decode, instruction execution, memory access, and write

back. A Complex Instruction Set Computing (CISC) architecture could have a much large number

of pipelines stages; for example, an Intel Pentium 4 processor has a 35-stage pipeline.

• Data parallelism or loop parallelism. The program loops can be processed in parallel.

• Task parallelism. The problem can be decomposed into tasks that can be carried out concurrently.

A widely used type of task parallelism is the Same Program Multiple Data (SPMD) paradigm. As

the name suggests, individual processors run the same program but on different segments of the

input data. Data dependencies cause different flows of control in individual tasks.

In 1966 Michael Flynn proposed a classification of computer architectures based on the number

of concurrent control/instruction and data streams: Single Instruction, Single Data (SISD), Single

Instruction, Multiple Data (SIMD), and (Multiple Instructions, Multiple Data (MIMD).3

The SIMD architecture supports vector processing. When an SIMD instruction is issued, the operations

on individual vector components are carried out concurrently. For example, to add two vectors (a1, a2, . . . , a50) and (b1, b2, . . . , b50), all 50 pairs of vector elements are added concurrently and all

the sums (ai + bi ), 1   i   50 are available at the same time.

The first use of SIMD instructions was in vector supercomputers such as the CDC Star-100 and the

Texas Instruments ASC in the early 1970s. Vector processing was especially popularized by Cray in

the 1970s and 1980s by attached vector processors such as those produced by the FPS (Floating Point

Systems), and by supercomputers such as the Thinking Machines CM-1 and CM-2. Sun Microsystems

introduced SIMD integer instructions in its VIS instruction set extensions in 1995 in its UltraSPARC

I microprocessor; the first widely deployed SIMD for gaming was Intel’s MMX extensions to the x86

architecture. IBM and Motorola then added AltiVec to the POWER architecture, and there have been

several extensions to the SIMD instruction sets for both architectures.

The desire to support real-time graphics with vectors of two, three, or four dimensions led to the

development of graphic processing units (GPUs). GPUs are very efficient at manipulating computer

graphics, and their highly parallel structures based on SIMD execution support parallel processing of

large blocks of data. GPUs produced by Intel, Nvidia, and AMD/ATI are used in embedded systems,

mobile phones, personal computers, workstations, and game consoles.

An MIMD architecture refers to a system with several processors that function asynchronously and

independently; at any time, different processors may be executing different instructions on different

data. The processors can share a common memory of an MIMD, and we distinguish several types of

systems: Uniform Memory Access (UMA), Cache Only Memory Access (COMA), and Non-Uniform

Memory Access (NUMA).

An MIMD system could have a distributed memory; in this case the processors and the memory

communicate with one another using an interconnection network, such as a hypercube, a 2D torus,

a 3D torus, an omega network, or another network topology. Today most supercomputers are MIMD

machines, and some use GPUs instead of traditional processors. Multicore processors with multiple

processing units are now ubiquitous.

Modern supercomputers derive their power from architecture and parallelism rather than the increase

of processor speed. The supercomputers of today consist of a very large number of processors and cores

communicating via very fast custom interconnects. In mid-2012 the most powerful supercomputer was

a Linux-based IBM Sequoia-BlueGene/Q system powered by Power BQC 16-core processors running at

1.6 GHz. The system, installed at Lawrence LivermoreNational Laboratory and called Jaguar, has a total

of 1,572,864 cores and 1,572,864 GB of memory, achieves a sustainable speed of 16.32 petaFLOPS,

and consumes 7.89MW of power.

More recently, a Cray XK7 system called Titan, installed at the Oak Ridge National Laboratory

(ORNL) in Tennessee, was coronated as the fastest supercomputer in the world. Titan has 560,640

processors, including 261,632 Nvidia K20x accelerator cores; it achieved a speed of 17.59 petaFLOPS

on the Linpack benchmark. Several most powerful systems listed in the “Top 500 supercomputers” (see

www.top500.org) are powered by the Nvidia 2050 GPU; three of the top 10 use an InfiniBand 4

interconnect.

The next natural step was triggered by advances in communication networks when low-latency and

high-bandwidth wide area networks (WANs) allowed individual systems, many of them multiprocessors, to be geographically separated. Large-scale distributed systems were first used for scientific and engineering

applications and took advantage of the advancements in system software, programming models,

tools, and algorithms developed for parallel processing.