DOI:10.1145/3492896

Vinton G. Cerf

On Heterogeneous Computing

ONE OF THE major challenges in the development of the Arpanet was solving the problem of communication between heterogeneous computers. In the late 1960s and 1970s, there were several computer makers and their machines had varying word lengths, binary coding schemes, instruction sets and a plethora of operating systems. The underlying homogeneous network of Interface Message Processors (IMPs), which we would call “routers” or “packet switches” today, offered a uniform interface to the heterogeneous “host” computers connected to the Arpanet. The Network Working Group, led by Stephen D. Crocker, solved the problem by the invention of the Network Control Protocol (NCP) and application protocols such as File Transfer and TELNET (remote terminal access). Coping with heterogeneity is a challenge. The Internet designers tackled the problem of interconnecting heterogeneous packet-switching networks using the TCP/IP Protocol Suite.

In the computing world, the Reduced Instruction Set Computing architecture (RISC) has provided widely adopted instruction set design principles for which David A. Patterson and John L. Hennessy received the prestigious 2017 ACM A.M. Turing Award. Although I am not a hardware designer, I have been struck by the observations of others such as Margaret Martonosi, Assistant Director of the National Science Foundation for Computer, Information Systems and Engineering and Google colleague, Robert Iannucci, that heterogeneity is returning to computer design with concomitant challenges for compiler designers. In addition to RISC-based CPUs, we now see Graphical Processing Units (GPUs), Tensor Flow Processing Units (TPUs), Quantum Processing Units

(QPUs), and Field Programmable Gate Arrays (FPGAs) in use or looming on the horizon. Each of these has unique properties that allow for optimal programming solutions to hard (and even NP-hard) problems.

The idea of using a mix of computing capability is by no means new. In the 1950s and early 1960s, my thesis advisor, Gerald Estrin, and his colleagues worked on what they called “Fixed Plus Variable Computing.”^a I have written about this before.^b This time I want to focus on the challenge for compiler writers to map conventional and new programming languages into functional operation on a variety of programming platforms, bearing in mind their various results and potential parallelism must be accounted for by the compiler. Martonosi points out that testing and analysis must be applied to increase confidence that the physical devices work as intended and that the mapping of a program onto the hardware mix

produces the intended computational result. Anyone familiar with the problem of numerical analysis will appreciate that details count. For example, loss of precision in large-scale floating-point computations can deliver erroneous results if inadequate attention is paid to the details of the actual computation.

Among many other considerations, a compiler writer will need to determine how data input or initial state is established for the computing unit in question. How will data be represented? How will it be advantageously transferred to other, heterogeneous computing components in the system? How will the flow of control of the computation be managed if parallel operation is anticipated? What will the “runtime” environment look like? If a computation goes awry, how can this be detected and signaled? In some sense, these are old questions demanding new answers in a more heterogeneous computing environment. Just as the Arpanet and Internet designers wrestled with interoperability, so must the designers of heterogeneous computing environments.

There is something simultaneously satisfying and unsurprising about these questions. Computing is an endless frontier in which we have an unending supply of new problems to confront in the search for new solutions. It is of vital importance to pursue these ideas. “Computational-X,” for virtually all scientific values of “X,” is part of a paradigm shift that began in the mid-20th century and continues unabated today. Our ability to compute on grander scales and in new ways will have significant influence on the rate at which scientific understanding progresses.

^a Estrin, G. Organization of computer systems—The fixed plus variable structure computer. In *Proceedings of the Western Joint Computer Conf.* (San Francisco, CA, USA, May 3–5, 1960).

^b Cerf, V.G. As we may think. *Commun. ACM* 58, 3 (Mar. 2015), 7.

Computing is an endless frontier in which we have an unending supply of new problems to confront in the search for new solutions.

Vinton G. Cerf is vice president and Chief Internet Evangelist at Google. He served as ACM president from 2012–2014.

Copyright held by author.