



Calhoun: The NPS Institutional Archive
DSpace Repository

NPS Scholarship

Publications

1993-10

On Massively Parallel Computers

Hamming, Richard W.

Monterey, California: Naval Postgraduate School

<https://hdl.handle.net/10945/63722>

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

Downloaded from NPS Archive: Calhoun



Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

<http://www.nps.edu/library>

mpc

October 28, 1993

ON MASSIVELY PARALLEL COMPUTERS

R. W. Hamming

There will be so many voices demanding that we must finance research and development of massively parallel computing (MPC) that my more modest remarks such as, "Perhaps we ought to consider financing work in the area." will have little effect so I pass on to a stronger recommendation.

First, I must divide up types of computers into special purpose and general purpose machines. The ENIAC as originally conceived was to solve only trajectory problems, but was soon converted painfully to general purpose computing. The weather problem can be seen as a special purpose computer, and there are others. I will call these special purpose machines Type 1 machines. Unfortunately many Type 1 machines are soon converted, as were the ENIAC and Whirlwind, to general purpose machines, Type 1a. On the other hand there are Type 2 machines, originally conceived as general purpose machines. Some of these, for example MARK 1 at Harvard, end up special purpose because of the inability to program them easily. Let these be Type 2a.

Except for Type 1 it is well known that these days software costs dominate hardware costs. This is a fundamental fact that the proposers of MPC want to forget, but we must keep them honest in this matter.

Software, in its turn, falls easily into two types, Type 1 which is the software utility programs, without which we cannot reasonably run a modern machine of Type 2. Type 2 software is quite different from much of programming which refers to preparing general problems for the machine.

I believe that Type 1 software has been done so long that it is reasonable to demand the type of control we apply to the hardware engineering; we should in fact demand that it meet engineering specifications. Of course when building the hardware of a new machine we may have to build one or more test parts to see if we can in fact do what we hope to do, and similarly in the software it may be necessary to build a piece here and there that is novel, but in the main the hardware is planned in detail before any of the final construction is done, and similarly I believe, we should demand that of the software. The software shall be laid out in detail such as, "using the well tested algorithm of ... we will assign the storage, we will find the parallelism in programs via ... algorithm, etc." The estimates of the final operation of the software should be similar in detail and reliability to the estimates for the hardware.

Every proposal for other than Type 1 machine shall have both the hardware and software details, and meet, at all stages, the similar engineering design criteria. Furthermore, nothing shall be finally built before both are completely designed to meet the similar engineering design standards.

If you stick to this perfectly reasonable standard then you will save a lot of money, hardware fanatics will have to find software support backing to parallel their MPC computer proposals, and software designers will, in this limited area of utility programming, Type 2, have to meet reasonable engineering standards.

Rule 1. Proposals for other than Type 1 machines must include software as well as hardware plans, including cost estimates that are realistically arrived at.

Rule 2: You cannot build anything, except test modules, until both the hardware and software are designed to the corresponding engineering standards.

I believe that the firm application of these two simple rules will save a lot of time, money, and wasted effort, and will in fact advance the field significantly in the long run. It will meet a lot of opposition from the hardware people, but I think they are wrong in their objections. History has already produced too many MPC machines which lay idle most of the time and were merely play things. These days it is probably easier to build hardware to fit the software that you can build, than it is to build the hardware and then try to fit in the software. We have made this mistake too often in the past, and if on an MPC project either of hardware or software is to be ahead of the other, then it must be the software that is ahead.

These rules about software are more important in the area of MPC than they are in the Type 2 machines because there is less that can be borrowed when a completely novel concept of hardware is proposed. I do not feel that I am being too strong in this area of the fundamental importance of software, but I am sure the rules will be highly resented by some. I believe that the adherence to them will in the long run advance the whole field of MPC, and nothing else that I can think of will do as much.