A New Day–A New Way
(A Challenge to the MPP Community)

Steven J. Beaty, Ph.D.
Hewlett-Packard
Fort Collins, Colorado U.S.A.
beaty@fc.hp.com

Gearold R. Johnson, Ph.D.
National Technological University
Fort Collins, Colorado U.S.A.
gerry@lance.colostate.edu

Abstract

People have been using MPP for all the wrong reasons. Most have used it for single-process speed improvement. This is not what we really need: we really need computers to be extensions of ourselves for them to be truly useful. They need to use their parallelism to be special purpose, redundant, and always busy. We need them to understand their environment, so we do not have to. We need them to understand our environment so they do what we want. We need to reward them when they do what we want.

Massively-parallel computers demand a new paradigm for programming. It is not enough that we simply use the same old sequential languages with parallel constructs superimposed. A new model must be developed that more closely mimics the way humans develop. In this paper we challenge the computer scientists to develop these new models for massively-parallel systems. One method is to use competing strategies with learning mechanisms superimposed. Such techniques were pioneered by John Holland at the University of Michigan. We propose the use of object-oriented functional languages and Holland's classifier techniques for the development of a learning operating system that controls a network of computers.

1 Introduction

We are very fond of a recent quote by Peter Drucker [3]

> Every few hundred years in Western history there occurs a sharp transformation. Within a few short decades, society rearranges itself – its world views; its basic values; its social and political structure; its arts; its key institutions. Fifty years later, there is a new world. And the people born then cannot even imagine the world in which their grandparents lived and into which their own parents were born. We are currently living through just such a transformation.

The current transformation is not associated with just computing. Transportation systems have been revolutionized during the past fifty years with large jumbo jetliners darting back and forth across the oceans at speeds unimaginable just fifty years ago. Instant satellite based communications systems allow us to sit in our living rooms and watch events unfurl half way around the world, even before those next door realize what is happening. Truly, everyone is now part of a 'global culture' whether they want to be or not. But, it is the field of computing that has accelerated the transition into the next world view with its profound societal changes. In this paper we will attempt to describe our view of what the computing world should look like on the other side of the current transformation, as Drucker calls it.

Computer hardware developments are now accelerating so quickly that it has become difficult even for those in the computing profession to stay abreast of the development. In the quote "good ol' days," new generation computing machinery was costly to develop, test, manufacture and sell. It is no accident that the IBM 360 [1] series was the generation of the decade of the 60's. It was followed by the 370 family a decade later, the 3080 family the following decade, and the 3090 series this decade. The 3090 family is driven by microprocessor chips, which raises issues we will address shortly. These machines came a decade apart in time. Machines had seven to ten year lifetimes. Life was far simpler.

Today, new generation hardware is introduced every twelve to eighteen months. Performance improvements are staggering. The costs for developing these systems are also staggering. New architectures with associated new fabrication facilities can cost over US $1 billion to construct. In the days when a good production of computers was in the thousands this would have been unheard of. Today with sales in the tens of millions, even these extraordinary costs can be recovered and the manufacturing of these devices is incredibly profitable for the companies able to make the investments.

It is also very clear that the single chip microprocessor has not achieved its physical performance limitation yet. Single chip microprocessors have at least three orders of

magnitude left in processing speed to be achieved, and perhaps even more. Historically, it has taken about ten years for supercomputer speeds to be obtained in a single chip microprocessor. For example, today's DEC 21064 [6] processor is comparable to the Cray 1 computer of a decade ago. Cray's final machine, the Cray-4 operated at 1000 MHz, nearly three times faster than DEC's Alpha running at 300 MHz, or six times faster than the Intel Pentium running at 166 MHz. Obviously, the Cray-4 speed will be achieved in future single chip processors. A 400 MHz PowerPC [2] chip is the design stage now. It appears that single chip processors will see clock speedups of a factor of 5 to 8 over the coming years.

Instruction level parallelism (ILP) will also offer a nearly ten fold increase. Currently, processors with four simultaneous execution units are available. Simulation indicates that ILP yields positive results for up to 12 to 16 execution units, perhaps even 30. ILP would thus seem to still offer an order of magnitude in processing speed.

Symmetric Multiprocessing (SMP) will add another order of magnitude to processing power in the next several years. A number of machines are currently available that easily handle four processors, and there is no reason to believe that ten processors will be any more difficult.

Additional on-chip facilities for branch prediction and speculative execution will also help reach these performance predictions. Pundits have long been saying we are approaching the single chip speed limit; we do not believe this to be true. Performance improvements in single chip microprocessors will provide performance improvements of 800-2000X over the next fifteen to twenty-five years. The machines with these performance characteristics will be cheaper than the current generation uniprocessor machines and far cheaper than special purpose hardware such as massively-parallel computers.

In fact, the design of general purpose massively-parallel computing is continually out of date. By the time the machines can be designed, software developed to make them useful, the inexorable march in the performance of single-chip processors makes the machine useless and obsolete. We must turn our attention to other avenues rather than the design of massively-parallel computing systems.

2 Why MPP?

Massively-parallel computing is not important for increasing computational speed. Sure, there are classes of problems that will always demand the fastest processing speed possible, but these problems will never be important enough to justify the software development processes needed to make such classes of computers general-purpose and therefore affordable to many users. The recent demise of Cray Computer Company (CCC) illustrates this very problem. Twenty years ago, special-purpose government research centers could justify the high cost of software development to make the supercomputer industry viable. This is no longer true. Therefore, supercomputers, or the latest manifestation–massively-parallel computer systems–must have another problem set to justify their design, construction, and use. Other manufacturers have learned this lesson. Thinking Machines is now a software-only company as is Kendall Square Research. Even Cray Research, Inc. has been purchased by Silicon Graphics, Inc.

While massively-parallel computing researchers were working on designing the next generation machine, the single processor personal computer and workstation world built the largest massively-parallel computing system imaginable – the Internet. It is already estimated that ten to thirty million computers are connected worldwide by the Internet. This translates into nearly 10^{16} instructions per second of potential execution speed. Couple this current state with the rate of increase and we potentially will see as many computers as the Earth's population (at least one per person) equaling the number of neurons in a human brain. We argue that even six billion computers may be a low estimate of the total number of interconnected machines. It is easy to envision a world where every member of the society has hundreds of computers working for them and each needs to be able to communicate with all those around it, or within its context, as we say.

Historically, it seems we keep doing the same problems over and over, just making them faster. What we really need to do is solve problems people care about, and not calculating more digits of π. It is not about speed, it is about doing the right thing. Humans are so much slower than computers. We need them to do the right thing for us with very little input from us.

Redundancy and robustness are more important than solving problems faster. We need to have our computers working all the time, sensing what we need them to be doing. When one is down, another should take its place. When everyone is asleep on one side of the world, use those machines to help the daylight side. This is easy to predict, even on a person-by-person basis.

What might work is special-purpose, general processors. Each could be very similar to all others. Each would have a different set of input and output devices and could therefore learn about its own environment and its tasks. A microwave, a refrigerator and a PC are not that much different. Each needs to translate inputs to outputs following functions. With the rise of commodity chips, each could have the same processor. This would allow for easy interchange of information as the network access scheme could be the same. Each could start with generic programming related to the task at hand, and then learn about its environment. Given goals, each could learn about how to achieve them. For ex-

ample, a person is likely to watch the same set of television shows each week. The TV and VCR could easily note when these events occur. What if the person is stuck in traffic on the way home and cannot tell the VCR to record a favorite show? Could not the VCR take the initiative and record the show itself? At worst, a little extra rewinding would be needed (with the VCR knowing the amount of course) and at best the human would enjoy a show that would otherwise be missed. If the TV and VCR talked with each other, this could be extended. What if the TV noticed that the human did not turn it on at a usual time and channel? The TV could notify the VCR to record the show. What if we include the person's workstation in this scenario? It would know that the person was working late and notify all the computers at home to do the right thing. Humans are fairly easy to predict, especially when combined with the fact that we are surrounded by many computers during an average day.

3 MPP is not Massively-Sequential Processing

We assert the MPP has not only failed to live up to its billing, but that in fact, it cannot. Very few people are using MPP on a regular basis; those that are, are using it on a very application-specific basis. This is because no one is using MPP as it should be used: to do things, things we have not been able to do, instead of doing the same old things faster. People are still stuck in the mode of thinking about computers in a serial fashion, as that is how the higher level of consciousness behaves. We are not aware of all the underlying processes going on in our heads, all of them very parallel. They occur at a level lower than speech (the basis of higher-level thought) in a massively-parallel fashion. We have too long modeled computation on speech and written communications and have not allowed computers to function at a level suitable for them. We have too long been afraid to let them form structures and produce actions that we do not completely understand, while this happens all the time between our own ears. We ask them to behave as very fast calculators, tying their hands for becoming something truly useful. We must ask ourselves:

1. what are we afraid of, and
2. what do we really want computers to do?

Do not we want them to increase the quality of our lives? Do not we want them to reduce the complexities of modern living? It seems we must not as most computers do the opposite.

So, we need to add more communication channels between processors. We need to try and think in a parallel fashion, something very unnatural to us. We need to allow the computers to act as neurons – lower-level units than we are used to, and allow them to form their own cooperating and co-operating structures. These needs are difficult to express in today's computer languages.

4 Classifier Systems

Holland, in his seminal work of 1975 introduced the concept of classifier systems [5]. Classifier systems (CS) encompass three subsystems: the classifier (a rule-based system), a reinforcement algorithm (Holland suggested a bucket brigade), and a new-rule discovery system (a genetic algorithm (GA)).

The classifier consists of a rather simple set of rules that are simply bit patterns (with "don't cares") and an input/output mechanism (See figure 1). At the input side, messages arriving from outside the system are converted into bit patterns that are then matched with the rule set of the classifier. Rules that "fire" produce outputs that are placed in a message list. At the completion of a run, all fired rules are translated into output messages and passed to the outside world through the output interface. Payoff information is returned to the input side. In this model, neither the input or the output interface are alterable by the system itself, a impediment to true learning.

The most difficult part of a classifier system is the development of explicit rules and a method of cascading payoff back to the rule sets most responsible for the overall system success. This is obvious because many times long sequences of activities must occur before the system knows whether or not it is successful. Keeping track of these sequences is inherently complex. Holland developed the bucket brigade model specifically for this application.

The genetic algorithm discovery system acts to replace rules using traditional GA techniques. The new rules could be refinements of existing rules or totally new rules through rule mutation.

5 MPP and Learning

So why would a computer want to learn about a human's actions and attempt to predict good behavior on that basis? What is the computer's motivation? It would seem that all learning must take place in an environment that provides rewards for good behaviors and punishments for bad behaviors. At first glance, it would seem difficult to motivate inanimate objects. The rewards however do not have to be tangible (as they often are not in humans). Many silicon reinforcement systems would probably work in learning computer systems. A simple scheme would be a add a reinforcement register to the machine. Each time a human rewarded a behavior, the register would be incremented. A

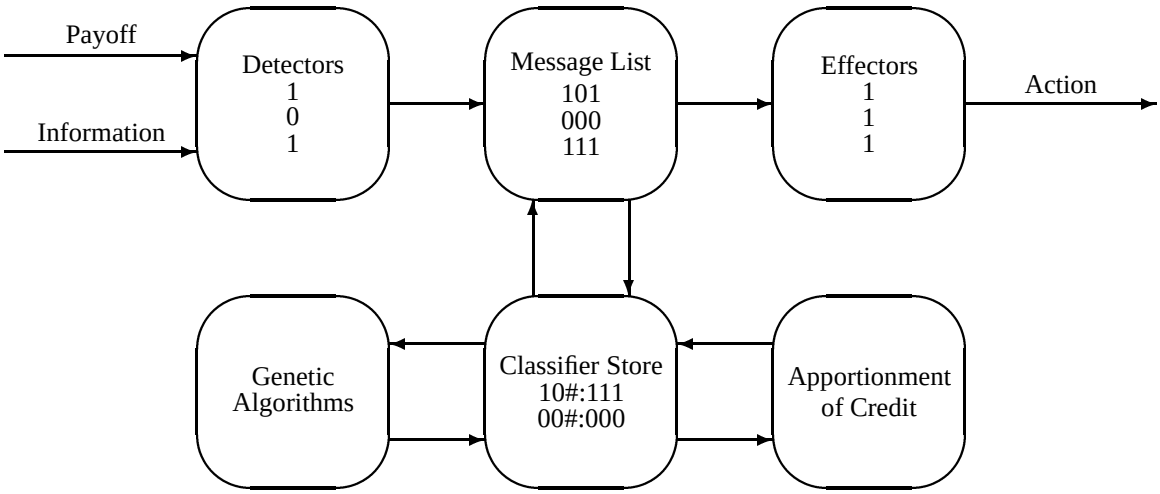

Figure 1: Schematic for a Classifier System

simple yes/no button as input to the machine could be the reward/punishment device. As the register value increased, the learned rules would be elevated in stature. While very simplistic in nature, this scheme has certainly driven learning in other systems. CS seem a natural match for this kind of learning.

For many years, we have written operating environments we have understood. At least, that the writer has hopefully understood. We then spend many years and many dollars attempting to ascertain what was meant by the writer of the program, instead of having the program understand what we are like. If a program is flexible and can observe its environment, it would be much better for it to learn about the human than vice versa. Humans are creatures of habit, probably more so than computers are. We tend to do the same act over and over, often repeating mistakes. Why do not word processors learn from their spell checkers what words we always misspell and fix them on the fly? Why do not VCRs learn we always fast-forward through commercials and do it for us? Why do not radios learn our listening habits and tune to the correct station for the five o'clock news? Fear of the unknown appears to be a major force. When something happens on a computer, even if we like what happens, we are fearful. If something good happens and we do not know why, why not reward the behavior and move on? This is what we do in the rest of our experience. If something bad occurs, we make the best of it and put devices in place so the event is not repeated. Why not give this power to computers who could make very good use of it?

It is clear that the early pioneers in machine learning understood that computers could learn about the things the computers themselves do. For example, David Goldberg [4] states that

> DeJong's work considered genetic algorithms in function optimization settings although he was well aware of the potential for GAs in other domains: he was especially interested in their application in data structure design, algorithm design, and computer operating system adaptive control.

This statement seems to have fallen on deaf ears for the literature is devoid of machine learning applications associated with the computing system itself rather than some problem domain application such as gas pipeline control.

Once a processor reaches a certain level of competency, it should communicate this to its neighbors. Perhaps when a certain value is reached in the reward register, or when the rules have stabilized to a certain level, the processor could be ready to take on similar tasks from other sources, or teach other processors how to do its job. All the TVs a person owns should really have about the same information about the person's habits. There is little reason for each to form the rules without the help of the others. In the same way, all compute power can be distributed. For example, let us assume a processor has learned how to put two numbers in order. Another has learned how to break down lists into their component parts. A system is asked to sort 100 numbers. The first processor states that this is far too many number for it to sort, and asks for a smaller list. The second is able to provide this, breaking the list down into manageable pieces for sorting. These two could then broadcast that they can sort larger lists together, and are rewarded for this.

6 Conclusions

In this paper, we suggest that MPP has been headed in the wrong direction. We believe that there is still a lot of computing power available in traditional technologies, and that there is plenty of compute power available on the world's networks. Performing traditional types of numerical compute tasks more quickly is neither interesting nor is it a valid direction for MPP. MPP gives us the opportunity to have a major advance in the usefulness computers provide to humans. The computers should be able to correctly predict our behaviors and desires. We should reward them for doing the right thing and we should all them to communicate with each in order to do truly useful tasks. We need to develop new languages and paradigms in order to facilitate this this new day in computers. We need to allow the computers (and networks) to learn about their environments using methods such as classifier systems. Maybe in these ways, MPP can finally fulfill its promise.

We would like to challenge the MPP community to try to make computers more useful to humans. We would like to see an example of a simple learning system based operating system or network operating system. Classifiers might be built that use either, or both, competitive systems or cooperating systems. It seems that both paradigms are used by humans to learn. The computing world offers a wonderful laboratory for trying different models for learning on constrained environments. It certainly could be more useful than the way we currently study learning in humans. And in the end, maybe they can tell us more about ourselves than we already know.

References

[1] D. Anderson, F. Sparacio, and R. Tomasulo. The IBM system/360 model 91: Machine philosophy and instruction-handling. *IBM Journal of Research and Development*, 11(1), Jan. 1967.

[2] K. Diefendorff. History of the PowerPC architecture. *Communications of the ACM*, 37(6):28–33, June 1994.

[3] P. F. Drucker. *Post-Capitalist Society*. Harper Business, 1993.

[4] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[5] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[6] R. L. Sites. Alpha AXP architecture. *Digital Technical Journal*, 4(4), 1992.