PYRAMID MULTI-COMPUTERS, AND EXTENSIONS AND AUGMENTATIONS

by

Leonard Uhr

Computer Sciences Technical Report #523

November 1983

PYRAMID MULTI-COMPUTERS, AND EXTENSIONS AND AUGMENTATIONS

by

Leonard Uhr

ABSTRACT

Pyramids appear to have local and global algorithm-structured architectures that will execute parallel algorithms with great speed and efficiency. But they cannot do everything. Therefore this paper proposes a variety of augmentations [(17) examines A-C G further]:

A) Integrate a Pyramid (or an Array) with a Host
B) Effect Different Instructions at Each Pyramid Layer
C) Assign M Controllers to an NxN Base Pyramid
D) Assign Two or More Computers to Each Node
E) Expand Pyramids into Lattices
F) Link D+1 D-Degree Pyramids at their Bases
G) Increase Power, Reconfigure Each Computer Moving Up
H) Set Reconfiguring Switches Between Sets of Processors
I) Sandwich Processor Layers Between Memory Layers
J) Augment a Pyramid to Form a Dense De Bruijn Network
K) Add Separate Broadcast and Message-Passing Abilities
L) Combine Pyramids with Networks of More Powerful Nodes

PYRAMID MULTI-COMPUTERS, AND EXTENSIONS AND AUGMENTATIONS

Leonard Uhr[1]
Department of Computer Sciences
University of Wisconsin-Madison

Pyramids appear to have local and global algorithm-structured architectures that will execute parallel algorithms with great speed and efficiency. But they cannot do everything. Therefore this paper proposes a variety of augmentations [(17) examines A-C G further]:

A) Integrate a Pyramid (or an Array) with a Host
B) Effect Different Instructions at Each Pyramid Layer
C) Assign M Controllers to an NxN Base Pyramid
D) Assign Two or More Computers to Each Node
E) Expand Pyramids into Lattices
F) Link D+1 D-Degree Pyramids at their Bases
G) Increase Power, Reconfigure Each Computer Moving Up
H) Set Reconfiguring Switches Between Sets of Processors
I) Sandwich Processor Layers Between Memory Layers
J) Augment a Pyramid to Form a Dense De Bruijn Network
K) Add Separate Broadcast and Message-Passing Abilities
L) Combine Pyramids with Networks of More Powerful Nodes

PARALLEL-PIPE PYRAMIDS; ARRAYS; SERIAL COMPUTERS

Pyramids and similar hierarchical, layered, converging multi-resolution multi-computer architectures overcome many of the problems of multi-computer arrays:

They offer a structure that gives far more efficient:

a) message-passing and convergence of data, reducing distances from (in an NxN array) O(N) to O(logN);

[1]Partially supported by NSF Grant MCS-8109103.

b) reduction, storage and processing of information, when the pyramid is used as a parallel pipeline through which images are flowed and transformed;

c) convergence together of globally distant information, which can now interact without message-passing;

d) parallel operations: the $O(N^2)$ speed-up of arrays over 1-CPU computers plus an additional O(logN) speed-up from pipelining.

e) execution speed where parallel local near-neighbor operations are common;

f) global structures for reducing data while processing (as when identifying successively more global wholes), or/and for expanding data (as in broadcasting).

But a pyramid standing alone is not as powerful or efficient as an "augmented pyramid" embedded in a larger architecture. This paper explores several alternative augmentations.

PYRAMIDS, ARRAYS, PIPELINES AND SERIAL COMPUTERS

There is an enormous variety of possibilities for the structure of the pyramid, e.g., any of the inorganic or organic crystal structures that nature has evolved. But the PIP ("Pyramid Image Processor") - the one pyramid actually being built (13, 10) and also all the pyramids proposed to date (5,14,15,16) have a much narrower range of structures. Therefore a pyramid will be defined more restrictively as a structure of arrays, starting with a base array and with each subsequent array suitably smaller.

Consider an NxN \`array\` with each computer linked to its 4 square, 8 square and diagonal, or 6 hexagonal neighbors. Typically N will be a multiple of a small integer like 2 or 3 (consider this integer the pyramid\`s \`convergence factor\`), and the next-smaller array\`s length, N\`, will be N/2 or N/3 in length.

The arrays that have been built - the 64x64 DAP (9), 96x96 CLIP4 (4), and 128x128 MPP (1) - operate in "SIMD" (6) ("Single Instruction Multiple Data stream") mode. Arrays 512x512 or 1024x1024 are desirable for real world images. They are expensive, but not impossibly so - see (16) for an examination that leads to estimates of $1 to $10 per computer.

The one actual pyramid being built by Boeing will have 1 controller and 7 array-layers: 64x64, 32x32, 16x16, 8x8, 4x4, 2x2, 1x1. Each computer (except in base and apex) is linked to 4 children, 8 siblings, and 1 parent.

Such a pyramid is less than 1/3d larger than the base array alone. Thus a pyramid is only marginally more expensive than an array; but if the base array is idle almost all of the pyramid is wasted.

A VARIETY OF TECHNIQUES FOR AUGMENTING A PYRAMID

A) Integrate a Pyramid (or an Array) with a Host

The first way to augment a pyramid (or an array) is to give it a `host` computer. This may seem too obvious to mention, and in a weak sense arrays have hosts. But serial hosts can serve important functions that today they do not.

When a pyramid (or array or other network) is not appropriate, the host can take over, if adequately integrated with the pyramid. In addition, the host can be used for a wide variety of purposes, including coding, editing, compiling and debugging programs, and controlling, monitoring and receiving key information from the pyramid.

How best to link a large array or pyramid with an appropriately powerful host is a major unsolved problem, since the potential flow of data between them can be immense (on the order of the image size, e.g., 1024x1024x24-bits). But there are several attractive possibilities:

1) A very wide bus, e.g. N-bits wide for an NxN array, to link the base array and the host;
2) The host linked only to the apex, or to higher-level arrays (now information must be passed up the pyramid, then to the host);
3) The host linked to both low and high levels and switchable to either;
4) Memory shared by both pyramid and host.

B) Effect Different Instructions at Each Layer of a Pyramid

The Pyramid Image Processor, in the spirit of the CLIP, DAP and MPP arrays, will have a single controller. So every computer at every layer executes the same instruction (on different data). A simple variant would execute a different instruction at each layer.

A complete controller might be built for each layer, or simpler, different instructions broadcast to each layer. Now programs could pipeline images through different processes at each layer. For example, one might want to average, difference, find gradients, detect edges, look for local features, compound or combine features into larger features, and so on, executing whatever operations were most appropriate at each layer moving up through the pyramid.

C) Assign M Controllers to an NxN Base Pyramid

Several controllers might be assigned to each layer of the pyramid; more generally, M controllers to N computers. E.g.:
One controller would give the basic SIMD pyramid.
LogN controllers, one assigned to each of the layers, would give the pyramid described in B) above.
9logN controllers might be assigned 9 to each layer, each to a sub-array 1/9th the size.
N controllers might be assigned, one to each processor.
Another variant would use switches to link each controller to the set of computers it could be programmed to control.

D) Assign Two or More Computers to Each Node in the Pyramid

There appear to be two major types of operations that pyramid algorithms carry out, at least those developed to date. Information is transformed, combined and/or passed up the pyramid; or information is broadcast down.

This suggests that two nodes might replace each node in the standard pyramid, one to transform and process information upward, the other to move information downward. The upward computers might be made more powerful than their downward twins, since they execute more powerful and more complex operations. Each might be specialized to serve its particular purposes, e.g., the downward computers might be simple broadcasting switches.

Possibly most attractive would be a partitionable computer one could allocate as needed to upward and downward flow.

This suggests the further extension of using 3, 4, ... or whatever number of computers at each node. This gives further speed-ups if several processes can be executed in parallel.

E) Expand Pyramids into Lattices

Many argue that at the higher layers of a pyramid the processors should be successively more powerful. A first approximation that has other desirable properties makes every layer as large as the base, and links all into a lattice that retains the desirable logN properties of a pyramid. it is not at all clear how best to effect this linkage. But one alternative is to treat the logN arrays as a shuffle network (11). A 3-D lattice will be NxNxlogN, with logN as many computers as its base.

Such a system could be partitioned to execute several programs at the same time. An important alternate use would be to model systems where information diverges outward at the same time that it is being reduced and converged inward, as in the brain.

An alternate lattice can be constructed with N rather than logN layers (i.e., a diagonally linked 3-dimensional cube with diameter N rather than logN). A mixed pyramid can be constructed, using each type of O(logN) or O(N) convergence at some or at all layers.

F) Link D+1 D-Degree Pyramids at their Bases

The base layer`s computers can be given as many links as the interior nodes. One attractive way of doing this was suggested by Storwick (12): take D+1 pyramids and link them at their buds. This multiplies the number of computers by D+1, giving a substantial increase in density without increasing diameter. Indeed (see 16) this gives a denser system than can be got by expanding a pyramid into a lattice using the shuffle network (11) or similar interconnection topologies. There are still other even denser constructions of this sort (16).

G) Increase Power and Reconfigure Each Computer Moving Upward

One might modify a pyramid to have suitably sized processors at each layer (e.g., 1-bit, 4-bit, 16-bit, 64-bit moving up an 8x8, 4x4, 2x2, 1x1 pyramid). Note that there is no need to keep an exact balance in terms of the number of bits-worth of processors at each layer. Rather, one might want to take off-the-shelf processor chips, for economy and simplicity, and use them wherever most appropriate.

Alternately, one might design and fabricate reconfigurable computers that could be used either as B 1-bit or 1 B-bit processors. This would make possible reconfiguring to a system of the sort described in D) above, since either one or several computers could be assigned, by reconfiguring, to each node.

H) Place Reconfiguring Switches Between Sets of Processors

Large pyramids will have too many individual processors to add general reconfiguring networks (which need NlogN switches) that can completely remap data. Nor does this appear necessary, since the pyramid has relatively good remapping capabilities. But it seems attractive to consider adding a small reconfiguring network to link sub-arrays or clusters.

I) Sandwich Processor Layers Between Memory Layers

One especially attractive alternative would separate the processors from the memories, as follows: Images would input to an NxN array of memories at the base of the pyramid. An array of N/CxN/C processors (C=Convergence) would each have CxC of these memories as children, and one memory as parent.

The image can now be input into a base of memories that is substantially larger than the first array of processors. Since such an image array will usually be only 8 or 24 bits deep, the memory designed to store it could be made quite large at little expense. Similarly, the number of links from each node is substantially reduced, since no links are needed to siblings.

Processors share memories with children in one direction and parents in the other direction. But if the flow of information is either up or down through the pyramid, all processors will be accessing either their child or their parent memories, without contention.

Such systems need no more time. Input operands can be fetched from child memories, operated on, and stored into parent (or back into child) memories in the same number of cycles as when processors use their own memories.

J) Augment a Pyramid to Form a Dense De Bruijn Network

A tree (and therefore a pyramid that is basically a tree) has diameter $2\log_D N$ (D = degree). This can be reduced to at best the Moore bound (2), which is $\log_D N$. This may appear to be a relatively small reduction, since both are O(logN); and possibly it is. But there has been a good bit of interest in reducing the diameter of a graph with a given set of nodes (see, e.g., 12, 8, 16), in order to pull multi-computer nodes closer together, to improve message-passing. Message-passing in today`s multi-computer networks is extremely slow (16).

Almost all of the nodes in a tree are buds, so reducing the diameter just a bit may well reduce these speeds considerably.

Imase and Itoh (7) showed that De Bruijn (3) "shift registers" are the densest graphs found to date, starting around roughly 15,000 nodes. De Bruijn graphs are trees with 2D-1 links added to their buds, and D-1 links added to internal nodes. Thus a binary tree is augmented with three links to each bud and one extra link to each internal node.

K) Add Independent Broadcast and Message-Passing Capabilities

A pyramid can be augmented with a network of switches to handle broadcasting and other message-passing activities outside the pyramid, in parallel with actual pyramid processing. For example, when a pyramid is used to pipeline and transform moving TV images input to the base every 30 milliseconds, an additional network might be devoted solely to downward message-passing.

Possibly simplest would be another pyramid, built by changing each node to two, with one a minimal switch - as in D) above. Or one might build a pyramid of switches and link each switch to a node within the pyramid of computers, but according to whatever linkage scheme desired, e.g., linking distant parts of the pyramid. A wide variety of linkages are possible and worth exploring. For example, one might want networks to remap, rotate, and broadcast.

L) Combine Pyramids with Networks of More Powerful Nodes

This final possibility includes more specific alternatives than all the previous augmentations combined, since basically it includes all possible bipartite graphs linking the nodes in a network of any sort and the nodes in the pyramid. I will simply mention a few potentially attractive examples and leave even the brief surveying of this type of augmentation to another place. More power at higher layers can be achieved, as described in G) above, by using successively more powerful processors moving upward within the pyramid, or by linking the higher-level pyramid nodes to more powerful processors in a separate closely-coupled network.

One of the simplest, and most attractive, of such schemes is to have an MIMD pyramid of more powerful processors linked to the higher layers of the large pyramid. For example a 4x4, 2x2, 1x1 pyramid of relatively powerful computers, e.g., 68000s, might be coupled by linking each 16-bit 68000 to 16

1-bit computers. Therefore the 4x4 array might be linked to the 256 computers in the 16x16 pyramid layer, etc.

More generally, with a pyramid of N-bit computers and an MIMD pyramid (or other network) of M-bit computers, we might link each M-bit computer to M/N N-bit computers. These might all be in the same or in different layers. They might be contiguous neighbors, or scattered widely. Thus, for example, when each N-bit computer is linked to both high-layer and low-layer M-bit computers it can serve not only to augment processing at the layers to which it is linked but also to message-pass information down and up the pyramid, at the same time processing this information.

An attractive alternative to the MIMD pyramid might be one of the optimally dense Moore graphs - either the 10-node degree=3, diameter=2 (10,3,2) Petersen graph or the (50,7,2) Singleton graph (2). For example, 9 computers might each link to (roughly) 1/9th of a higher-level layer and also to judiciously scattered lower-level processors, the 10th serving as back-up and spare. Or 9 computers could link to the 9 3x3 sub-arrays in a lower layer, the 10th to one or more computers in a higher layer. The Singleton graph can be similarly linked.

Any network topology that one might choose on whatever grounds, e.g., an N-cube, star, snowflake, X-tree or whatever (16), might similarly be linked to the large pyramid. But this brings up a wide variety of possibilities and of issues, and can only be pointed at here.

SUMMARY DISCUSSION

This paper describes pyramids, and explores why they have good algorithm-structured architectures for perception programs, especially those that attempt to recognize moving objects in real time. Then it suggests and very briefly examines a wide variety of possible augmentations to pyramids. Although appropriate for many perceptual tasks, pyramids alone will be relatively poor at some. And a perception program must ultimately be embedded in a larger intelligent system.

This paper has examined 12 possible augmentations of pyramids. Some make more powerful the pyramid`s nodes or layers (17); others add links, switches or/and computers within; others outside the pyramid:

A) Integrate the pyramid with a 1-cpu host that codes, edits, and compiles programs, and also executes processes for which the pyramid is poor.
B) Give each layer of the pyramid its own controller, so that all layers can work in parallel.
C) Give each sub-array or each sub-region of the pyramid a separate controller, so that different procedures can be executed at different regions and abstractions of the image. (More generally, assign C controllers to an N-computer network, where 0<C<=N.)
D) Replace each inner computer by two computers, so that information can be transform-piped both up and down the pyramid at the same time (or, alternately, both computers can work in parallel in the same direction). Extending this, use computers that can be partitioned to give the appropriate power in each of the two directions.
E) Expand the pyramid so that each array is as large as the base array (call this a `lattice`).
F) Link D+1 pyramids at their bases, to improve density.
G) Increase the size and power of each computer, moving up from base to apex, possibly also making the processors reconfigurable.
H) Add a reconfiguring network of switches between layers, sub-arrays, or/and individual processors.
I) Alternate processor layers and memory layers, so processors in adjacent layers pass information via shared memories.
J) Add links to the pyramid to form a De Bruijn network (asymptotically the densest structure discovered to date).
K) Add a network of switches to handle message-passing outside and in parallel with the pyramid.
L) Combine the pyramid with a larger multi-computer network.

These by no means exhaust the possibilities for augmenting and in other ways improving upon pyramids. But they include a number of interesting variations that appear to be well worth detailed investigation. And since most of these augmentations are not specific to pyramids or arrays they are candidates for making more general other more or less specialized structures.

REFERENCES

1. Batcher, K.E., "Architecture of a massively parallel processor," Proc. 7th Ann. Symp. Computer Arch. 168-174 (1980).
2. Bondy, J.A. and Murty, U.S.R., "Graph Theory with Applications," Elsevier, New York, 1976.
3. De Bruijn, D.G., "A combinatorial problem," Koninklijke Nederlandsche Academie van wetenschappen et Amsterdam, Proceedings of the Section of Sciences 49, 7:758-764 (1964).
4. Duff, M. J. B., "CLIP4: a large scale integrated circuit array parallel processor," Proc. IJCPR-3 4:728-733 (1976).
5. Dyer, C.R., "Pyramid algorithms and machines," In: "Multi-Computers and Image Processing," K. Preston, Jr. and L. Uhr (Eds.), pp. 409-420. Academic Press, New York, 1982.
6. Flynn, M "Some computer organizations and their effectiveness," IEEE Trans. Computers 21:948-960 (1972).
7. Imase, M. and Itoh, M., "Design to minimize diameter on building-block network," IEEE Trans. Computers 30:439-442 (1981).
8. Leland, W., Li Qiao and Uhr, L., "Globally dense (d,k) graphs for computer network architectures," Computer Sci. Dept. Tech. Rept., Univ. of Wisconsin, 1980.
9. Reddaway, S.F., "DAP - a flexible number cruncher," Proc. 1978 LASL Workshop on Vector and Parallel Processors, Los Alamos, 233-234 (1978).
10. Snapp, W., personal communication (1982).
11. Stone, H.S., "Parallel processing with the perfect shuffle," IEEE Trans. Computers 20:153-161 (1971).
12. Storwick, R.M., "Improved construction for (d,k) graphs," IEEE Trans. Computers (Short Notes) 19:1214-1216 (1970).
13. Tanimoto, S.L., personal communication (1982).
14. Tanimoto, S.L., "Programming techniques for hierarchical parallel image processors," In: "Multi-Computers and Image Processing," K. Preston, Jr. and L. Uhr (Eds.), pp. 421-429. Academic Press, New York, 1982.
15. Uhr, L., "Converging pyramids of arrays," Proc. Workshop on Computer Architecture for Pattern Analysis and Image Data Base Management, IEEE Computer Society Press, 31-34 (1981).
16. Uhr, L., "Algorithm-Structured Computer Arrays and Networks: Parallel Architectures for Perception and Modelling," Academic Press, New York, 1983, in press.
17. Uhr, L., "Pyramid Multi-Computer Structures, and Augmented Pyramids," In: "Computing Structures for Image Processing," (M. Duff, Ed.), Academic Press, London, 1983, in press.