Smart optoelectronic networks for multiprocessors

B. Layet[a], I. Gourlay[b], P. Dew[b] and J. F. Snowdon[a]

[a]Dept. of Physics, Heriot-Watt University, Edinburgh, EH14 4AS, U. K.

[b]School of Computer Studies, Leeds University, Leeds, LS2 9JT, U. K.

ABSTRACT

An intelligent interconnection network with fine-grain parallelism is described that has the potential to support efficient, scalable algorithms running on associated coarse-grain processors. The approach is relevant to the emerging computational cluster systems. The support of concurrent operations within the network is discussed and their mapping onto smart-pixel array network interfaces is shown. Choices are considered in the design of the free-space optical interconnect that enables the inter-smart-pixel-array communication. In particular, a system that uses multi-element macro-lenses is studied and results of detailed modeling are given that quantify the smart-pixel density. These results are used, in an illustrative case study of the sorting problem, to compare potential system architectures. This is work in progress and throughout the paper, important issues in the design and use of the intelligent interconnect are raised that require more study.

Keywords: Computational clusters, parallel computers, concurrent operations, smart pixels, free-space optics, optoelectronic networks, intelligent interconnect.

1. INTRODUCTION

An interesting and significant development in high performance computing has been the emergence of computational clusters[1-3], constructed from off-the-shelf PCs or workstations. For example the Beowolf project used Intel based machines running Linux[1], while Southampton University have investigated clustered DEC Alpha workstations running Windows NT[2,3]. A variety of interconnects have been considered[3]. The utilization of low cost commodity components offers cost effective high performance, when dynamic memory management is not important. Many larger machines, such as the SGI Origin2000[4] (which has a ccNUMA architecture) use data migration to support a virtual shared memory while reducing communication overheads.

This paper builds on the computational cluster model to study the design of an intelligent high performance network. By integrating smart-pixels[6] into the system it is possible to build an intelligent interconnect that can perform concurrent operations such as read&add[7]. This enables exploitation of fine-grain parallelism at the network level, leaving the processors (PCs) to perform the coarse-grain computation. This type of approach has the potential to support efficient, scalable algorithms that make use of (for example) a concurrent queue[7]. The concurrent operations required for algorithms like the queue are implemented on the intelligent interconnect, leaving the PCs (possibly dual processor) to perform the coarse-grain parts of the algorithm. It is expected that a standard PC interface would be used in the interface between the PCs and the smart-pixel based network. This set-up is shown schematically in Figure 1.

In addition to questions concerning the practicalities of constructing such a system there are a number of other interesting research questions. This paper considers whether efficient computations can be identified or designed that exploit the particular features of this arrangement and, also, the most appropriate way to implement the smart-pixel based intelligent interconnect.

In order to address these questions and highlight other important issues, concurrent operations and their implementation on a number of topological architectures is considered. In addition, an illustrative example, based on the well-documented problem of sorting[13], is presented.

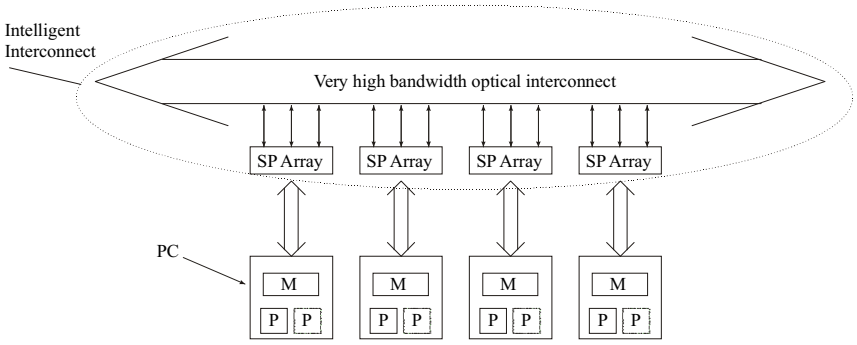

Figure 1: The envisaged scheme, where an intelligent interconnect is constructed using smart-pixels, with a high bandwidth interconnect, e.g. an Optical Highway[5].

The paper reports on this work (which is still in progress) and is organised as follows: In Section 2, the intelligent interconnect architecture is discussed. The type of computation that the network is expected to support is outlined, incorporating a brief review of concurrent operations. In addition, details of the physical architecture are described. Section 3 discusses details of the physical set-up used to implement the intelligent interconnect. Physical constraints imposed by various architectural models are considered, addressing the problems associated with mapping the logical structure of the algorithm on to a practical physical system. In Section 4, indications of key issues in assessing the set-up are addressed by means of an illustrative example based on a tree-structured sorting algorithm.

2. SYSTEM ARCHITECTURE

2.1. NETWORK LEVEL COMPUTATION

Before discussing details of the intelligent interconnect, it is appropriate to consider the type of computation it is expected to support. Essentially, the processors perform any required coarse-grain computations while (where possible) fine-grain parallelism is performed at the network level. The intelligent network incorporates a number of interconnected fine-grain processors (smart-pixels), the topology of which is discussed in the following sub-section. This is shown schematically in Figure 2.

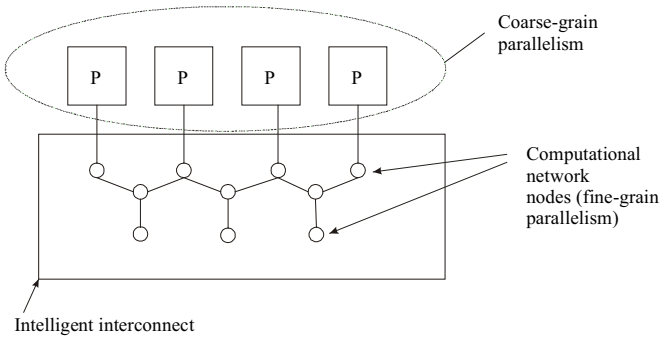

Figure 2: Schematic of the computational architecture. The fine-grain computational nodes are constructed using smart-pixels.

The intelligent interconnect is conceptualized as an array of integer processors that can perform concurrent operations. Hence the interconnect is capable of supporting the fine-grain parallelism required in concurrent algorithms such as the concurrent queue[7]. To clarify this, a brief review of concurrent operations is now given.

Suppose an algorithm requires the (concurrent) assignment of a unique array index to p processors and consider switches that can add two inputs, as shown in Figure 3. Two processors concurrently send inputs (A and B) to the switch. In addition, a global variable G is sent to the switch. The value (A+B) is output from the switch and the value A is stored. When the global variable value arrives, it is sent out on the left line and (G+A) is sent out on the right line. The global variable value is updated to (G+A+B).

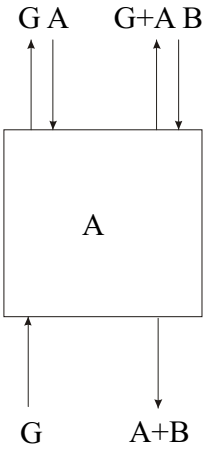

Figure 3: A read&add switch. A and B are the initial input values and G is the initial value of the global variable.

By forming a tree network of these switches, it is possible to perform the array index assignment concurrently. This is simply a combining network for the concurrent read&add operation. For p processors this task can be performed in $O(\log p)$ time, with a tree network of depth $\log p$. To illustrate this, an example (with 8 processors) is given in Figure 4. Each processor inputs the value 1 to the combining network and a global input variable is initialized to 0. The effect of the combining network is that each processor retrieves a unique index from 0 to 7 and the global variable is updated to 8. More details on concurrent operations can be found in[7].

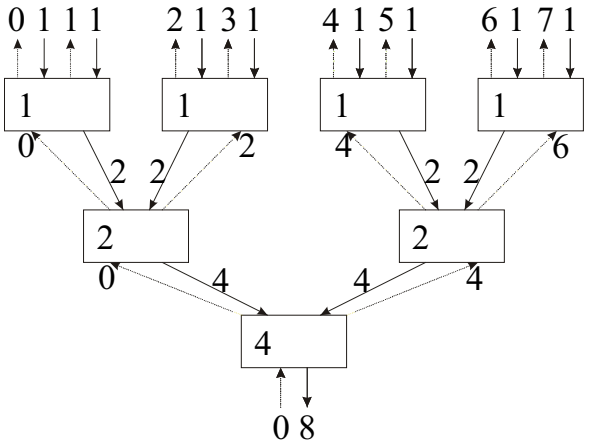

Figure 4: A combining network, performing concurrent read&add in order to assign unique indices to eight processors (the processors are not shown).

In order to be capable of supporting such operations, the interconnect needs to be capable of emulating a tree structure (see Figure 4), therefore the choice of interconnect architecture can be expected to

significantly affect the overall system performance. A number of logical topologies for the smart-pixel interconnect are discussed in the following sub-section.

2.2. INTERCONNECT ARCHITECTURE

This sub-section focuses on architectural issues relating to the intelligent network shown in Figure 1. It is assumed that the smart pixels are electrically connected in an on-chip mesh, whereas off-chip connections to other smart-pixel arrays (SPA) are supported by the optical interconnect. This paper concentrates on architectures in which every smart pixel in a particular array is connected to a correspondingly located smart pixel on one or more other arrays. This is not necessarily an optimal strategy, but is has the advantage of conceptual simplicity. Note that it permits all the smart pixels in an array to transmit a unit of data to another array (to which it is directly optically connected) in O(1) time. The smart pixel processing capabilities are sufficient to implement a combining switch such as the read&add switch shown in Figure 3.

A key architectural choice is the topology of the (SPA) interconnection network. The optical system (see Section 3) is assumed to be linear, since it is possible to embed any desired logical topology within this linear physical topology. This paper considers the following logical topologies: the linear array, the mesh and the hypercube logical topologies. Consider, as an example, an 8-node hypercube. In this network every node is connected to 3 others. For example, node 0 is connected to nodes 1, 2 and 4. In this context a node is an SPA. This means that it is possible to perform simultaneous transmission of a unit of data from every smart pixel (SP) in the array at node 0 to any of these other nodes; SP_j on SPA_0 communicates with SP_j on SPA_i, where $0 \leq j < p$, p is the number of pixels in an array, and (in the hypercube example) $i = 1$, 2 or 4. The generic architecture of the fine-grain layer is indicated in Figure 5.

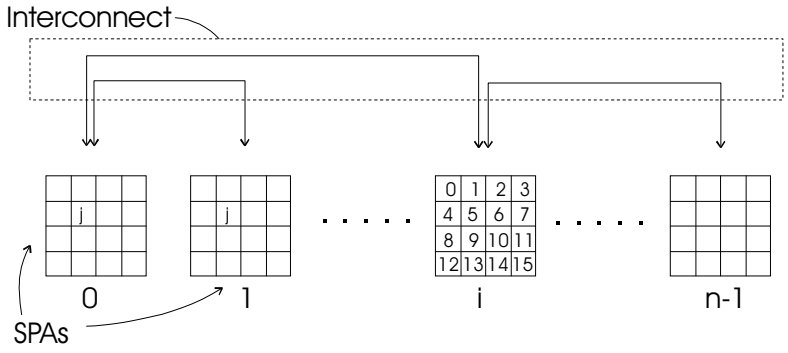

Figure 5: Fine-grain architecture in a system with n arrays of $p = 16$ smart pixels each, showing an arbitrary selection of optical connections.

It is necessary to consider how the tree structure (see figure 2) can be mapped onto this physical system. The mapping is dictated by the requirement that every main processor must have access to a leaf node in order to inject its (potentially) concurrent memory request into the tree structure. An additional requirement is that that the root of the tree has a direct link to the main processor whose local memory can satisfy the requests. Therefore, a mapping is chosen in which every level of the tree is flattened onto a single layer of SPAs. Consider the example shown in Figure 6. The 16 main processors (each interfaced to the network through an SPA) with the memory to be accessed residing on processor 0. The squares represent the SPAs and the lines that join them show the communication pattern. Since the vertical axis indicates the flow of time it is clear that the same 16 SPAs are shown in each row. A bold square indicates that the SPA is active at that stage in the tree, whereas a fainter square indicates an inactive SPA. The horizontal axis indicates the processor to which the SPA is attached. It is assumed that the processors are indexed corresponding to their order in the optical system.

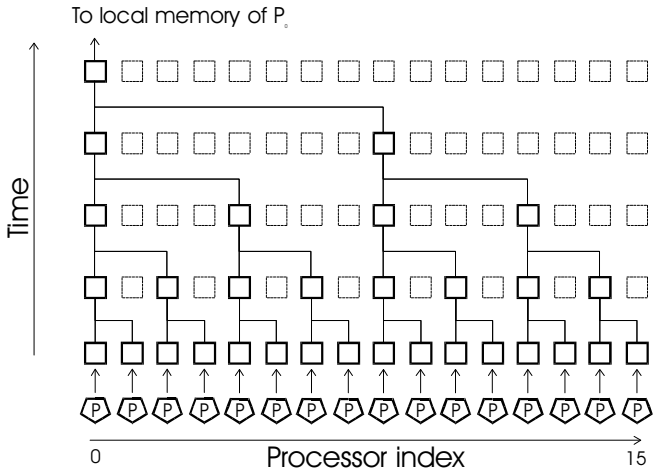

Figure 6: Time evolution of SPA usage during a concurrent operation (or similar tree-structured computation). Bold squares correspond to active SPAs. The main processors are shown at the bottom of the diagram.

Consider the simultaneous execution by all main processors of a concurrent operation, accessing the same global variable, located in the local memory of processor 0. Each processor sends an appropriate message to the network, as shown in Figure 6. Hence, initially, every SPA will hold one message. To reach the next level up, adjacent pairs communicate their messages to one member of the pair and they are combined. This process continues with the SPAs that are not empty, with the number of active SPAs halving at each stage. The number of stages until a single active SPA (the one on the destination processor) is therefore $\log n$. The concurrent operations require a combining switch such as the read&add switch shown in Figure 3. A value must be stored for every combination that takes place. Since the SPA that eventually becomes the root takes part in $\log n$ combines it must store $\log n$ values in addition to the value passed to memory. So, as the system grows the storage requirement increases but only as $\log n$. (This places a limit on the smart pixel density in an array, in addition to optoelectronic-transducer-based limit derived later, in Section 3.)

The number of communication steps, L, required for the combination onto a single SPA depends upon the network topology. For the linear array, mesh and hypercube topologies these are given by

$$L_{LA} = 2\sum_{i=0}^{\log n - 1} 2^i = n - 1 \tag{1}$$

$$L_{M} = 2\sum_{i=0}^{\frac{\log n}{2} - 1} 2^i = 2\left(\sqrt{n} - 1\right) \tag{2}$$

$$L_{HC} = \log n \tag{3}$$

In the preceding discussion the nodes in the tree structure were referred to as SPAs. Of course, it is actually the smart pixels within the SPAs that form the nodes. The discussion and the results remain valid if one assumes that each square in Figure 6 represents a single smart pixel, and they are located in identical positions on different arrays (so that communication between smart pixels within a single array is unnecessary). Since each array can consist of many smart pixels it is easy to see that multiple trees, with different root nodes, can coexist in the system. Hence, many concurrent operations directed at different variables in the local memories of any main processor can be dealt with simultaneously.

The above discussion assumes that all smart pixels are in corresponding locations in their different arrays. This may not be simple to achieve in practice. One (partial) solution is to allocate the same part of each SPA to trees that are rooted in a particular node. The main processor sending a request to the network then at least knows what part of the SPA to use, if not the exact smart pixel, hence on-chip communication is minimized (off-chip communication is unaffected).

Having identified possible architectures it is desirable to make some detailed comparisons. This is partially addressed by the methods introduced here and in Section 4, although it largely remains an area for future work.

A model of communication costs in the interconnect is required. Based on the assumption of cut-through routing (of which wormhole routing is a subset), the communication time for a message of m words, that traverses l links in the interconnect is[8]

$$t_{comm} = t_s + lt_h + mt_w \tag{4}$$

where t_s is the message startup time, t_h is the per-hop time (or node latency) and t_w is the per-word transfer time. The total communication time for an operation that uses N stages of communication with the same message size in each is then

$$T_{comm} = Nt_s + Lt_h + mNt_w \tag{5}$$

where L is the total number of links traversed. The relative values of the constants are difficult to quantify at this stage. The message start-up time is generally assumed to be large because of the software overheads and protocols, however, within the network the overhead will be much smaller, perhaps only several times the node latency. The per-hop time is generally quite small, but may be increased in an optoelectronic architecture by the electrical-optical and optical-electrical conversions that must take place. Two cases are examined here; in one the per-word transfer time is equal to the per-hop time, and in the other the per-word transfer time is three times as long.

Finally the communication time to combine n synchronous concurrent operations can be considered. For the three topologies under consideration, the total number of links traversed in the combining network is given by Equations 1-3. Substituting these expressions in Equation 5 and assuming that m = 2 (since the messages need only contain an address and an integer) the graphs in Figure 7 can be produced. It is seen that higher connectivity networks provide increased speed, although the improvement of the hypercube over the mesh is minimal, and, unsurprisingly, the benefit is only significant for larger networks. If, in addition, the potentially large start-up time for a message to *enter* the network is considered the distinction between the topologies is further reduced. A key question is therefore: how much communication will occur in the intelligent interconnect that is not dependent on frequent synchronization with the coarse-grain processors? This remains to be addressed in future work.

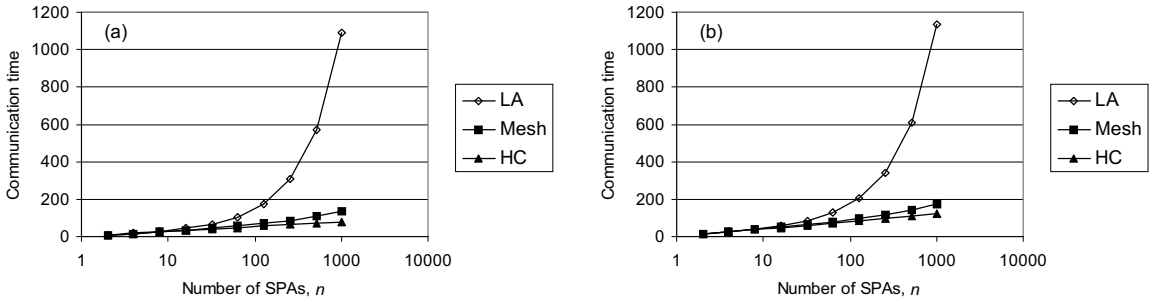

Figure 7: Communication time to complete a concurrent combination. The communication constants are: (a) t_s = 5, t_w = 1 and t_h = 1 (b) t_s = 5, t_w = 3 and t_h = 1.

In practice, a penalty is paid to obtain a higher connectivity. In the next section we examine some details of the physical implementation of the intelligent interconnect, and show how the smart pixel density must be traded off against connectivity.

3. IMPLEMENTATION OF THE SMART-PIXEL INTERCONNECT

The previous section introduced a smart-pixel architecture and showed how it can support concurrent operations. A number of topologies were considered. A free-space optical interconnect that enables the inter-SPA communication to support these topologies is now described and some implementation issues are considered.

Telecentric *4f* image relays have often been used in demonstrator systems to optically connect planes of optoelectronic devices,[9] although technologies such as the fiber image guide provide alternatives. The former option is considered here. In Figure 8, two SPAs are shown with associated optics, within an example system that allows each SPA to communicate with its neighbor to the right. Patterned mirrors permit optical power to be sent to modulators on the SPAs from laser sources above. The arrangement of polarization beam splitters (PBS) and wave-plates permits reflected beams from the modulators to be directed to a neighboring SPA. The system is unidirectional and direct optical links are only available between nearest-neighbors. A bi-directional system can be created by combining two such systems, but data now arrives at a node in two distinct, spatially separated areas. This may be significant in creating an electrical transmission bottleneck, particularly if highly parallel fine-grain processing on the SPAs is envisaged, in which case high bandwidth data must be received, processed and re-transmitted by the SPAs.

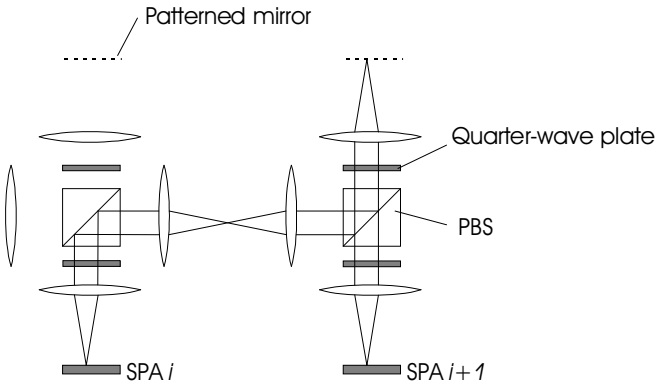

Figure 8: Optical system for connecting SPAs. A beam of light reflected from a modulator (optical power beam not shown) leaves SPA *i*, is reflected at two PBSs and a patterned mirror, and is then (having had its polarization rotated by a round trip through a quarter-wave plate) transmitted by the second PBS to be focussed on SPA *i+1*. The same optical path is followed by a VCSEL beam generated on SPA *i*.

A potential solution makes use of polarization-controlled VCSELs as on-SPA optical sources.[10] Referring again to Figure 8, the quarter-wave plates below the PBSs are no longer required, and the mirrors are no longer required to be patterned. The polarization orientation of a VCSEL determines whether the beam of light it generates is sent to the neighboring SPA on the left or on the right. (It takes 3 image relays in either case). Links in either direction can thus be distributed over the area of the SPA and the potential bottleneck is averted. It must be noted that the VCSEL technology to enable this solution is still being developed.

Another alteration to the system permits direct optical links to be made between non-adjacent SPAs. The addition of patterned half-wave plates in the image planes between the PBSs allows the selective retardation of some of the optical beams. The polarization of a beam from SPA *i*, for example, can be rotated so that it is not deflected by the PBS above SPA *i+1* (or by other PBSs further to the right in the system). At another inter-PBS image plane, further along the system, the polarization is rotated again so that the beam is deflected by the next PBS it meets and is coupled onto the associated SPA. This technique is applicable in both modulator-based and VCSEL-based systems. In the latter case, bi-directional communication is possible. It can be enabled in a modulator-based system by using a component in the inter-PBS image plane that is both a patterned mirror and a patterned wave-plate.

Only some of the optical systems just described are suitable to implement the interconnection architectures of Section 2.2. The optics must provide bi-directional links from each SPA and, also, in order to implement

the higher connectivity versions, direct communication between non-adjacent SPAs. Thus, a VCSEL-based system with inter-PBS patterned half-wave plates or a modulator-based system with patterned mirror-wave-plates are suitable. The linear array, mesh and hypercube SPA interconnection topologies can be embedded in these (physically) linear optical systems. This has been discussed elsewhere.[5,11]

In optical interconnects of the type assumed here, a physically distinct point-to-point link is provided for each connection that is supported. This is required for a full implementation of the networks, and it avoids optical power budget problems associated with fan-out. Clearly, in this case, the number of optoelectronic transducers in a smart pixel must increase with the connectivity of the network. Thus, the increased connectivity of the mesh and the hypercube compared to the linear array comes at a price. In fact, not only are more transducers required but, in addition, the detectors must be larger because the signals travel through more image relay stages to connect non-adjacent SPAs and the focussed spot size is increased by the accumulated aberrations. Further, there is an enforced sparseness of transducers on the SPA due to signals that pass through the relay optics but which do not go down onto the SPA. The corresponding area on the SPA cannot be used for transducers (at least not without arbitration) since a conflict would arise with the existing connection.

The Code V ray-tracing package (from Optical Research Associates, Pasadena, CA) has been used to model the spot size obtained from cascaded five-element macro-optic relay lenses[12] with the assumption of reasonable fabrication tolerances. Alignment tolerances of the lenses with respect to each other, on the other hand, are not modeled. The alignment is assumed to be perfect. In this case, distortion and magnification of the spot array is minimal and there are no difficulties in the registration of the spots to detectors arrays. The significant measure of performance is the spot size. The spot size determines the feasible smart pixel density.

Using these results, the number of smart pixels in an array can be calculated under the assumptions that the field of the lens is 1 cm^2, and the optical links are serial. Figure 9 shows the number of smart pixels per array, p, that are possible for various values of n, the number of SPAs. A significant penalty is paid in system size (in terms of the total no. of smart pixels, np) to obtain greater connectivity. Note that parallel links would reduce the number of smart pixels according to the level of parallelism required.

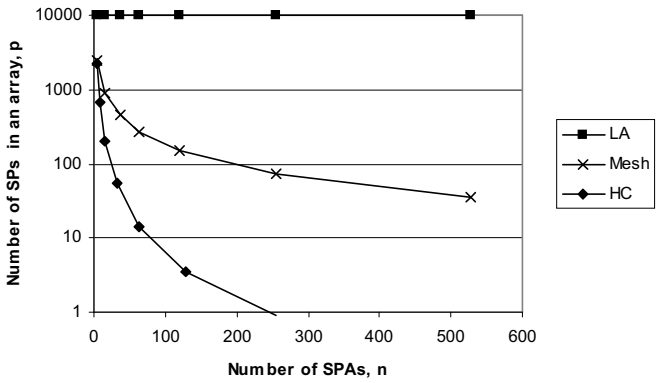

Figure 9: Maximum number of smart pixels in an array as a function of total number of arrays, for the three topologies considered.

4. SORTING

The mapping of the combining network that is required for concurrent operations onto the intelligent interconnect was considered in Section 2.2. To take the study further, and in particular to assess the smart pixel density vs. connectivity trade-off that has been identified in Section 3, it is useful to consider a specific algorithm. Considered here is a tree sorting which maps well the tree structure necessary for the concurrent operations. Future studies will extend the range of algorithms considered. Before proceeding further, the tree sort is briefly introduced.

Suppose we have a tree network, consisting of log n leaf processors (which could be PCs) where n is the number of data elements to be sorted. The remainder of the network consists of $(\log n - 1)$ computational nodes (residing within the intelligent network), with the sorted elements being passed from the root node into memory in ascending order, as the algorithm proceeds. The sequence of n elements is initially distributed across the leaf processors, which locally sort their elements using an optimal sequential sort. The computational nodes connected to these processors then merge sequences received and pass the new sorted sequences up the tree. This process continues until the fully sorted sequence resides at the root. The sequence is then stored in memory (e.g. on a main processor). An example of this is shown in Figure 10. More details of this algorithm can be found in.[13]

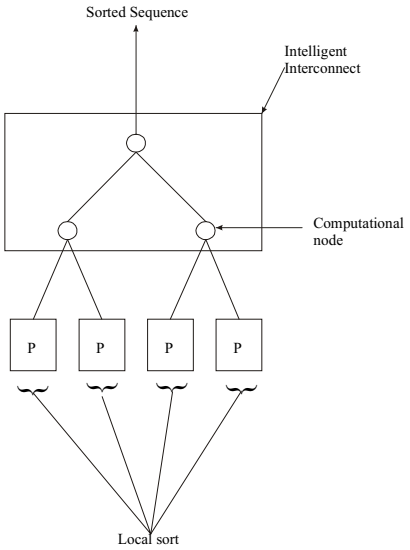

Figure 10: Example of a tree network for sorting. Each main processor (PC) performs a local sort and the sorted subsequences are passed (concurrently) to the intelligent interconnect where the sort is completed.

The communication costs of the algorithm are considered for the three topologies discussed earlier: the linear array, mesh and hypercube. If a single smart pixel is allocated to each node in the hypercube then the communication time must be evaluated with a message size, m, equal to the subsequence size. However, in the other topologies more smart pixels are available and the subsequences may be communicated with a degree of parallelism, and a lower value of m can be used. The relative values of m can be determined directly from Figure 9. The communication time in the tree sort is largely determined by the link between the two SPAs that have the root as their parent (because these are furthest apart in the physical optical system, and most stages in the algorithm use this link), so the time to make this communication is simply compared using Equation 4. The results are shown in Figure 11 for two sets of values of the communication constants. The hypercube architecture performs badly in both cases due to its low smart pixel density. The mesh only begins to challenge the linear array if the per-word transfer time falls close to the per-hop time. However, no firm conclusions on the optimal topology can be drawn as yet. In particular, the hypercube can be defended by the assertion that a problem (or an algorithm) was not chosen in which it could demonstrate its benefits. Also, a large smart-pixel size may be dictated by the silicon area required to

implement its processing function, in which case, the optical system may not the limiting factor in the smart pixel density and a higher connectivity architecture will be appropriate.

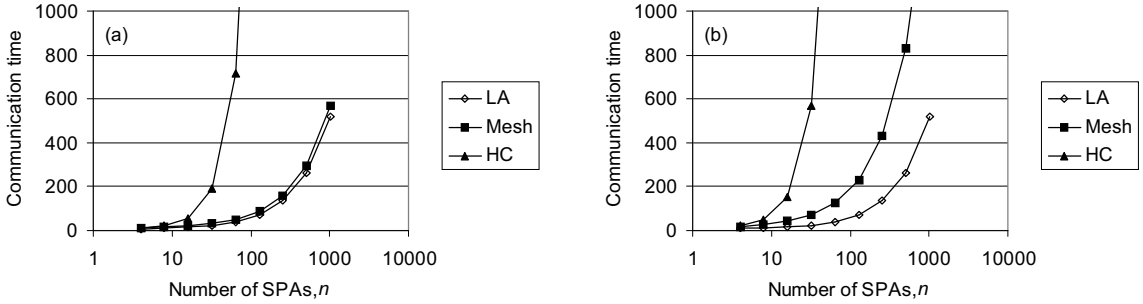

Figure 11: Communication time in tree sort. The communication constants are: (a) $t_s = 5$, $t_w = 1$ and $t_h = 1$, (b) $t_s = 5$, $t_w = 3$, $t_h = 1$.

5. SUMMARY AND DISCUSSION

A preliminary study has been made of an intelligent interconnect with potential applications in high performance computing using computational clusters. The interconnect is intended to perform fine-grain parallelism at the network level, allowing the main processors to perform coarse-grain computation. An optoelectronic implementation based on smart pixels has been considered. The cost and commercial novelty of the smart-pixel based technology is currently an obstacle, but this will not change until the usefulness of the technology is clearly demonstrated.

A key task is to identify how concurrent operations may be supported by the interconnect. In this paper a mapping of the combining network required for concurrent operations (which has a tree structure) onto an assembly of optically interconnected smart-pixel arrays has been shown. However, there are still many important questions to be tackled in future work. For example, this paper has not addressed the communication between the coarse-grain processing node and the fine-grain smart-pixel array. Is there a potential bottleneck here? Also, how do concurrent operations executed at different nodes find each other within the fine-grain layer?

The simulated performance in terms of the i/o density in an area, of a suitable free-space optical system has been reported for three topological variants of the intelligent interconnect. The modeling assumes realistic multi-element macro-lens fabrication and perfect alignment of these lenses. The results quantify the trade-off between smart pixel density (as limited by optical i/o) and network connectivity.

Whilst lower connectivity topologies are preferable because of the high smart pixel density that can be achieved, the time to perform a concurrent memory access is increased compared to the higher connectivity topologies. A sorting algorithm that maps onto the tree structure has been used to assess the appropriate balance. It is found that the hypercube topology performs poorly due to the low smart pixel density. The mesh and the linear array topologies give a somewhat improved performance, although this depends upon the communication model used. The optimum choice remains a subject for further study. An important next step is to examine the performance of an algorithm that explicitly uses concurrent operations (such as the priority queue). Design of new concurrent algorithms is also useful, to broaden the scope of such analysis. It should also be mentioned that the communication model needs to be considered in more detail.

Ultimately, the practicality of the smart optoelectronic network that has been proposed depends upon technological factors and the details of the physical implementation. So far, the optical imaging system has been considered in the greatest depth, and practical issues of other aspects of the implementation have been largely ignored. In further work, the capabilities of the smart pixels (e.g. memory and processing capacity) and the optoelectronic devices within them should be incorporated. Of course, simple models are required to enable further study of the intelligent interconnect at a reasonably high level.

REFERENCES

1. The Beowulf Project, http://beowulf.gsfc.nasa.gov/
2. D. A. Nicole, K. Takeda, I. C. Wolton, "HPC on DEC Alphas and Windows NT", Proc. HPCI '98, Manchester, pp. 551-557, 1998.
3. S. J. Cox, D. A. Nicole, K. Takeda, "Commodity High Performance Computing at Commodity Prices", WOTUG-21 Proc. 21st World Ocean and Transputer User Group Technical Meeting, Canterbury, 1998.
4. SGI Origin 2000, http://www.sgi.com/origin/2000
5. J. A. B. Dines, J. F. Snowdon, M. P. Y. Desmulliez, D. B. Barsky, A. V. Shafarenko, and C. R. Jesshope, "Optical interconnectivity in a scalable data-parallel system", *J. Parallel and Distributed Computing* **41**, pp. 120-130, 1997.
6. T. M. Pinkston, C. Kuznia, "Smart-pixel-based network interface chip", *Appl. Opt.* **36**, pp. 4871-4880, 1997.
7. J. M. Nash, "A study of the XPRAM model for parallel computing", PhD thesis (1993).
8. V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to Parallel Computing*, Benjamin/Cummings, Redwood City, CA., 1994.
9. F. B. McCormick, T. J. Cloonan, A. L. Lentine, J. M. Sasian, R. L. Morrison, M. G. Beckman, S. L. Walker, M. J. Wojcik, S. J. Hinterlong, R. J. Crisci, R. A. Novotny, and H. S. Hinton, "Five-stage free-space optical switching network with field-effect transistor self-electro-optic-effect-device smart-pixel arrays", *Appl. Opt.* **33**, pp. 1601-1618, 1994.
10. N. Nishiyama, A. Mizutani, N. Hatori, M. Arai, F. Koyama, and K. Iga, "Lasing characteristics of InGaAs-GaAs polarization controlled vertical-cavity surface-emitting laser grown on GaAs (311) B substrate", *IEEE J. Sel. Topics in Quantum Electronics* **5**, pp. 530-536, 1999.
11. T. H. Szymanski, and H. Scott Hinton, "Reconfigurable intelligent optical backplane for parallel computing and communications", *Appl. Opt.* **35**, pp. 1253-1268, 1996.
12. D. T. Neilson, S. M. Prince, D. A. Baillie, and F. A. P. Tooley, "Optical design of a 1024-channel free-space sorting demonstrator", *Appl. Opt.* **36**, pp. 9243-9252, 1997.
13. S. G. Akl 'Parallel Sorting', Academic Press, 1985.