Topology Configuration in Hybrid EPS/OCS Interconnects

Konstantinos Christodoulopoulos, Marco Ruffini, Donal O'Mahony,
School of Computer Science and Statistics, Trinity College Dublin, Ireland
Kostas Katrinis, IBM Research, Ireland
christok@tcd.ie

Abstract. We consider a hybrid Electronic Packet Switched (EPS) and Optical Circuit Switched (OCS) interconnection network (IN) for future HPC and DC systems. Given the point-to-point communication graph of an application, we present a heuristic algorithm that partitions logical parallel tasks to compute resources and configures the (reconfigurable) optical part of the hybrid IN to efficiently serve point-to-point communication. We measure the performance of a hybrid IN employing the proposed algorithm using real workloads, as well as extrapolated traffic, and compare it against application mapping on conventional fixed, electronic-only INs based on toroidal topologies.

Keywords: Reconfigurable interconnection networks, optical circuit switching, communication graph, application mapping, partitioning, topology configuration.

1 Introduction

High Performance Computing (HPC) systems and datacenters (DCs) are being built with ever-increasing numbers of processors. Currently systems with tens of thousands of servers have already been reported to be in operation, while their scale is expected to grow to the order of millions of cores towards Exascale [1]. To obtain high system efficiency, computation and communication performance need to be balanced. Given the aggressive increase in compute density – thanks to the increasing number of cores/node and the growing deployment of accelerators – it is of paramount importance to avoid having the interconnection network (IN) become a bottleneck [1-2]; instead, IN technologies and system software need to grow in hand with the evolution in compute density to enable next generation HPC and DC systems.

Flagship supercomputers typically employ regular topologies of electronic switches, such as hypercubes and toroidal structures. For instance, the Cray XT5 [3] utilizes a 3D torus topology, while the K supercomputer [4] employs a 6D torus (although not all dimensions are complete). Such low-degree regular topologies are adopted due to their inherent ability to scale linearly with the number of compute nodes. Still, these sparse topologies tend –for specific applications– to aggravate the mapping of application communication to the underlying IN [5-10]. At the far end, many HPC clusters and DCs adopt indirect routing IN, such as fat-trees [11] that provide for full-bisection bandwidth and thus simplify the mapping of applications. Though, this comes at a cost that scales super-linearly with the size and thus raises scalability concerns when considered at extreme scale, especially if the investment is not justified due to poor utilization [12]. Oversubscription [13] is often proposed as a means of controlling the capital and power-consumption cost. Still, the inflexibility of placing network capacity once and for all at design time remains, similar to the case of low-degree regular topologies.

To close the gap between constant-degree, fixed INs and costly, full-bisection INs, past work [14-17] has proposed building low-degree, reconfigurable INs that allow for on-demand bandwidth allocation wherever is needed. Maintaining a low-degree IN reduces its contribution to the capital cost of the system, while these designs have been shown to fit well the

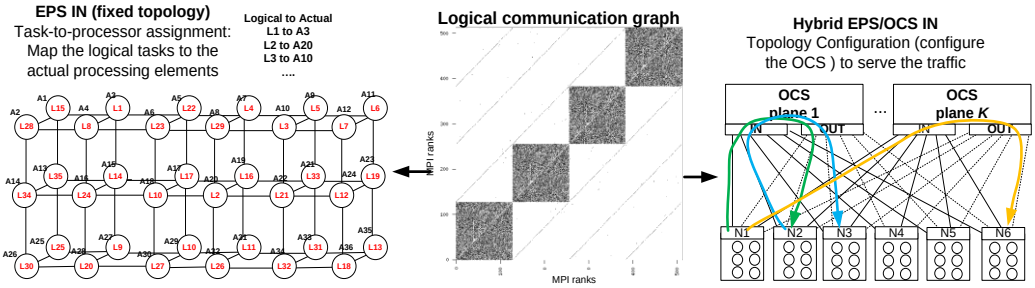

Fig. 1. Topology-aware mapping on fixed electronic-only IN (left) and topology configuration in reconfigurable OCS IN (right).

communication patterns exhibited by specific classes of HPC applications [14] and DC workloads [16-18]. The proposed reconfigurable IN architecture constitutes hybrids of two switching technologies, namely using both electronic packet-switches (EPS part) and optical circuit-switches (OCS part). The OCS part, typically implemented with commodity MEMS-based switches, handles high-rate, long-lived point-to-point flows, whereby the EPS part serves low-rate signaling, short-lived flows and collectives (many-to-many communication).

In this paper, we focus on traditional HPC applications that exhibit static point-to-point logical communication graphs. We call static an application that follows specific communication patterns so that its logical communication graph at intermediate phases and the aggregated graph at the end of its execution have well defined and consistent structures irrespective of the input. This definition is very close to the definition given in [14], where a static application is considered to have known communication pattern at compilation time. It must be noted here that logical point-to-point communication graphs are influenced only by application-level communication, i.e. point-to-point communication between logical entities, and thus do not depend on machine or IN characteristics. Work on identifying and classifying the point-to-point communication graphs of static HPC applications include [18] and [19]. Finally, stream computing applications [15] can be also included in this category of static applications.

Given the IN of an HPC system, mapping the tasks of a parallel application onto physical processors to allow efficient communication becomes critical to performance. This problem is usually referred to as *topology-aware mapping* and has been shown to be NP-hard [5]. The most efficient known approach [6] is to adapt the application source code to optimize task-to-task communication against a specific IN topology. Albeit efficient, this approach is cumbersome, while also being application- and system-specific, thus precluding re-use across machines or applications. More practical alternatives [7-10] do not require any changes to application code and thus trade-off efficiency for being application-agnostic. In particular, specialized system software takes as input the logical point-to-point communication graph of a static application and performs task-to-processor assignment taking into account the IN topology.

Among the various advantages [14-17] brought by reconfigurable INs, the ability to look at task partitioning/mapping from a different angle remains unexplored: instead of performing a sophisticated task-to-processor assignment on a fixed topology as in [5-10], the logical tasks can be partitioned and the topology of the (reconfigurable) IN that interconnects them can be configured for optimized communication and thus improved application performance. This comprises the topic of this paper. Fig. 1 contrasts these two different approaches.

Our methods apply and improve use cases that are relevant in mapping of static applications on fixed INs [5-10]. The logical point-to-point communication graph of a static application is identified at compilation time and/or the application is executed once to profile and capture its communication graph. Using this input, an optimized OCS topology configuration is computed and used to speedup subsequent executions of the application. We assume that the

OCS network is configured once, that is, it is configured at the outset of application execution and remains the same throughout its execution. Although the reconfigurable network could dynamically adapt its topology to support more efficiently the different application phases, we will not consider such cases here. References [16-17] examine the dynamic reconfiguration of the OCS network to follow the traffic variations of DC applications, considering only single-hop transmissions and mostly single-layer optical networks, features that we plan to extend in our future work. Note that dynamic reconfiguration can be viewed as a sequence of well defined static instances, whereby each one can be solved by the method proposed here.

For the purpose of evaluation, we profile several static HPC application kernels run on an HPC cluster using IPM [20] and derive their logical point-to-point communication graphs. Using *hop-bytes*[1] [8-10] as the performance metric and for various architectural choices of the hybrid target EPS/OCS IN, the level at which optical interconnection occurs and the number of optical ports available, we evaluate the performance of the proposed joint partitioning and topology-configuration heuristic and compare it against application mapping on conventional fixed, electronic-only INs that are based on toroidal topologies. We also develop simple models for estimating the capital cost of the hybrid EPS/OCS vs. a torus-like electronic-only IN.

2 System Model

We consider a generic multi-rack system architecture in which processing elements are multi-core processor chips. A given number of chips is mounted to a (compute) node. In turn, a number of nodes comprises a mid-plane, and a set of mid-planes is installed in a rack.

We assume an interconnection network (IN) adhering to a hybrid architecture comprising both an Electronic Packet Switched (EPS) and an Optical Circuit Switched (OCS) network [14]. The OCS network is typically implemented with one or more Micro Electro-Mechanical Systems (MEMS) optical switches (crossbar). MEMS are layer-0 switches that establish an end-to-end optical connection by reflecting a light beam from an input to an output port. The signal is switched transparently without performing any processing on switched data. MEMS-based switches exhibit circuit setup times that are typically in the order of tens of milliseconds and thus would result in prohibitively high per-packet switching overhead, should they be operated as packet switches. Therefore, it makes sense to use MEMS as switching elements of high-rate, long-lived, point-to-point flows. Flows at core- or chip-level do not currently have such characteristics. Moreover, since MEMS-switches support solely point-to-point communication, creating a chip-level network would require a high-degree OCS topology, which is prohibitive in terms of scalability. Therefore, all-optical switching is applied to aggregated traffic at a higher interconnect level that we hereafter refer to as the *optical aggregation level.* Given current cores/node figures and typical byte/flops ratios, setting the optical aggregation at mid-plane or rack level maximizes the utilization of optical circuits and reduces the frequency of reconfiguration of OCS-switches (due to aggregation). Still, as compute density packed into a single node keeps increasing, so will do the number of compute tasks and bandwidth per node [1], justifying the placement of optical aggregation at the node level. To allow our work to capture this trend and thus be future-proof, we apply an abstraction to the assumed system model to enable us to carry parametric studies.

Specifically, we logically cluster processing elements (cores) together to form groups that we refer to as *logical clusters* (LCs). The processing elements comprising an LC are interconnected via an electronic packet switched (EPS) network (to be referred to as the first-level of EPS network or EPS-edge) that serves both intra-LC communication, as well as aggregating

[1] The hop-bytes metric is defined as the weighted sum of traversed hops. Weights correspond to message sizes.

LC traffic destined to distant LCs. At the optical aggregation level, *M* EPS (bidirectional) ports are connected to the second-level of the EPS network (EPS-core) and also *K* (bidirectional) ports are connected to the OCS network. At the optical aggregation level the traffic towards distant LCs can be served by two parallel networks: either the EPS-core or the OCS network. The OCS network handles persistent point-to-point, high-rate inter-LC flows, while the EPS-core network handles lower bandwidth and collective communications, as well as bursty flows. INs with two levels of hierarchy are also found in fixed topology EPS systems [10], and in the hybrid EPS/OCS IN [14-17]. While the above abstraction enables us to evaluate our approach against various choices of placing the optical aggregation level, it still allows a straightforward mapping of our approach to real systems. To showcase this, we assume an HPC cluster with both levels of the EPS network being implemented using Ethernet. The optical aggregation point is then either a multi-port Ethernet NIC (node-level optical aggregation) or an Ethernet top-of-rack switch (rack or mid-plane level optical aggregation), whereby optical transceivers are used to carry packets to/from the OCS network. The transmit (resp. receive) side of each transceiver is connected to an input (resp. output) port of the MEMS-switch.

Let *T* denote the number of processing elements that are grouped together to form a logical cluster (LC) and *N* denote the number of LCs of the system. In the hybrid EPS/OCS network model we adopt, each LC is connected to *K* parallel OCS planes at the optical aggregation level [14]. Fig. 2 depicts the architectural diagram of the system model we adopt in this paper. Since, each parallel OCS plane is implemented with an *N*x*N* crossbar switch, *K* *N*x*N* crossbar switches are required in total. A large MEMS crossbar switch can be configured to create smaller *N*x*N* crossbar switches, and so the total number of switches required can be less than *K*. Currently 320x320 MEMS crossbar switches are available by a number of vendors, while there are prototypes with 1000 in/out ports. Note that our goal is to create a hybrid EPS/OCS network with $K << N$ to keep the cost as low as possible and aim at massive scale out.

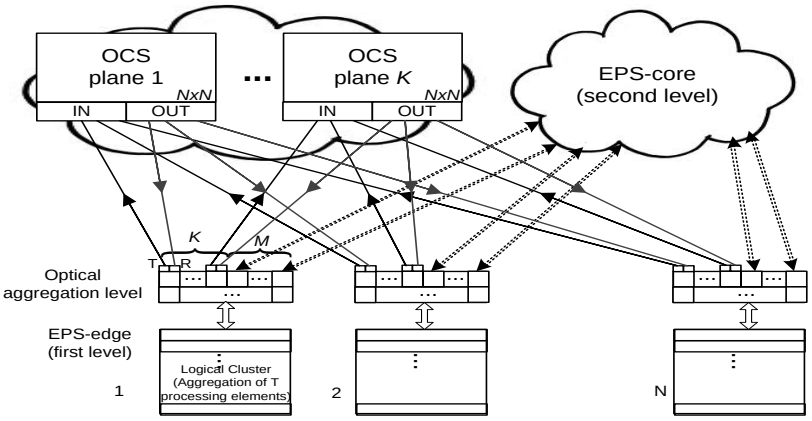

Fig. 2. Reference hybrid EPS/OCS IN consisting of *N* logical clusters (LCs). At the optical aggregation level, *M* and *K* bidirectional EPS ports are connected to the EPS-core and the OCS network, respectively. At each of the *K* EPS ports accessing the OCS network, the transmit (T) side of the transceiver is connected to an IN (input) port and the receive (R) side to an OUT (output) port of the OCS switch.

In this paper we will focus mainly on the OCS part of the hybrid IN. The method that we propose in the next section takes as input the logical point-to-point communication graph of a parallel application tasks, given e.g. in the form of a traffic matrix, and a specific OCS architecture as defined by the related *K* and *T* parameters. We assume that the rest of the traffic that is not point-to-point (e.g. collectives) is routed over the EPS-core part of the hybrid IN. The goal of the proposed method is to serve the point-to-point transmissions in an efficient way. To

do so, we partition the tasks to form logical clusters (LCs) and also derive the configuration of the OCS network to optimize the communication between LCs. The partitioning process divides the communication into inter- and intra-LCs communication, similar to the hierarchical mapping problem examined in [10]. Intra-LC communication that is served by the first-level, EPS-edge is considered to be *cheap* and is neglected, while inter-LC communication, which is routed over the OCS network, is the traffic that is optimized by our topology-configuration algorithms. In the remainder of this paper we will call this problem the ***Partitioning and Topology-Configuration with Bounded Connectivity*** (PTCBC) problem.

Given the adopted OCS IN with K parallel OCS planes, an application that after the grouping to LCs has an inter-LC communication graph with directed connectivity degree less than K, can be served over the OCS network with direct point-to-point connections. Otherwise, we resort to multi-hop transmissions. Multi-hop refers to OCS-based communication that involves electronic processing of packets at the EPS-edge, beyond the source/destination LCs, at intermediate LC *hops*. Multi-hop increases the effective bandwidth between LCs at the expense of increased latency as well as increased congestion and average network load.

3 Partitioning and Topology Configuration with Bounded Connectivity

In this section we formulate the ***Partitioning and Topology-configuration with Bounded Connectivity*** (PTCBC) problem and provide for an efficient heuristic algorithm to solve it.

We start with a parallel application that utilizes Z (MPI) tasks. A typical allocation would consist of assigning each task to a processing element corresponding to a processor core, although other assignments at the thread or chip level are also applicable. We are also given the logical point-to-point communication graph of the application (MPI task level communication) in the form of a traffic matrix $\boldsymbol{\Lambda}$ of size ZxZ. Element Λ_{nm} ($1 \le n,m \le Z$) corresponds to the point-to-point communication volume (in bytes) exchanged throughout the entire application execution between task n and m; thus $\boldsymbol{\Lambda}$ has a zero diagonal. The remaining transmissions that are not point-to-point (e.g. collectives) are routed over the EPS-core part of the hybrid IN and are not considered here. Tasks are first partitioned into logical clusters (LCs, see Section 2 for the definition of LCs). In particular, we assume that T tasks are grouped together to form the LC (one of the LCs may contain less than T elements, if Z mod $T > 0$). Let N stand for the number of LCs formed after clustering the Z tasks; then $N = \lceil Z / T \rceil$.

By partitioning into logical clusters (LCs) we transform $\boldsymbol{\Lambda}$ into a new traffic matrix $\boldsymbol{\lambda}$ of size NxN, with each element λ^{ij} ($1 \le i,j \le N$) corresponding to the point-to-point communication volume (in bytes) between LC_i and LC_j (i.e. $\boldsymbol{\lambda}$ corresponds to the inter-LC communication graph). Note that unlike $\boldsymbol{\Lambda}$ being part of the problem input, the traffic matrix $\boldsymbol{\lambda}$ is conditioned on the clustering method employed and as such is an intermediate output.

Inter-LC traffic, as described by traffic matrix $\boldsymbol{\lambda}$, is served by the reconfigurable OCS interconnection network (IN). The LCs are mapped to the physical compute resources to form compute aggregations (racks, mid-planes or nodes), by putting tasks that belong to the same LC together in the same compute aggregation without considering the position of the aggregation in the IN topology, since the OCS IN will be configured around these aggregations. As outlined in Section 2, the OCS network we consider consists of K parallel OCS planes. There are two different versions of the problem: one that assumes unidirectional and one that assumes bidirectional connections over the OCS network. This is related to the type/configuration of aggregation switches, the related transceivers used at the optical aggregation level and their connection to the MEMS switch(es). Given that the unidirectional case is more generic and that Ethernet commodity switches can support this operating mode, we

will focus on unidirectional connections, although our algorithms can be applied with minor changes to bidirectional connections. Note that if traffic matrix $\boldsymbol{\lambda}$ includes both directions of communication for an LC pair, the solution will include connections for both directions, although they may utilize different paths.

We start with a fully connected graph $\boldsymbol{G}=(\boldsymbol{V},\boldsymbol{E})$, with $\boldsymbol{V}$ being the set of vertices that correspond to the logical clusters (LCs) and $\boldsymbol{E}$ the set of candidate links that connect every pair in $\boldsymbol{V}$. The links included in $\boldsymbol{E}$ are not actually functioning but are the candidate links that can be established in the OCS network. In this context, we will say that we establish an optical link (i,j) when we configure the OCS network to establish a circuit connection between LC_i and LC_j. Since we are considering directed connections, the order of i and j is important. The number of links that can be established is constrained by the number K of parallel OCS planes. We let $\boldsymbol{L} \subseteq \boldsymbol{E}$ be the set of links that are chosen to be established and $\boldsymbol{G'}=(\boldsymbol{V},\boldsymbol{L})$ the related graph, with connectivity degree less or equal to K. $\boldsymbol{G'}$ is the graph of the constructed OCS network.

The Partitioning and Topology-Configuration with Bounded Connectivity (PTCBC) problem is defined as follows. We are given the traffic matrix $\boldsymbol{\Lambda}$, the desired number T of tasks per LC and the number K of parallel OCS planes, and we seek to identify a "good" partitioning of tasks to LCs (i.e. find the $\boldsymbol{\lambda}$ matrix), and a "good" configuration of the OCS network that interconnects the LCs (i.e. identify the set $\boldsymbol{L}$ of optical links to be established). The goodness of a joint partitioning/topology-configuration solution is measured against its ability to minimize a communication objective, namely *average hop-bytes*. The average hop-bytes metric is defined as the sum of path lengths (measured in hop-count) taken by the messages, weighted by the respective message size [8-10]. The purpose of the average hop-bytes metric is to capture approximately the average load on the network. In early works in the fields of graph embedding and VLSI design, emphasis was placed on the maximum dilation metric, which is the longest path (resp. the longest wire in a circuit). Ref. [8-10] argue that reducing the longest path (maximum dilation) is not as critical as reducing the average hops across all message sizes, as captured by the hop-bytes metric. Our algorithmic formulations are general and can be used to optimize other performance metric, such as the maximum dilation or link congestion, but the hop-bytes was chosen as the most appropriate metric for our comparisons.

If the optical aggregation level is placed directly at the processing elements, then partitioning to logical clusters becomes obsolete. This is a special case of the PTCBC problem with $T=1$ and thus $\boldsymbol{\lambda}=\boldsymbol{\Lambda}$. In this case, the problem is reduced to the topology-configuration problem with bounded connectivity degree, which we proved to be NP-hard by a reduction to the circular arrangement problem [21] (the proof is omitted due to space limitations). This case covers also the version of the problem where we are given directly the traffic matrix $\boldsymbol{\lambda}$ capturing the traffic that will be routed over the OCS network. Since the general problem described above with $T \geq 1$ includes as a special case an NP-hard problem, the general problem is also NP-hard. The optimal solution of PTCBC is bound to give at least as good performance results as the application mapping on hierarchical electronic-only fixed INs [10], as long as the connectivity degree of the hybrid network is not less than that of the fixed IN. For example, assuming that the optimal architecture to serve the traffic is a 3D torus, the optimal PTCBC solution would be to configure the OCS network to form a 3D torus and obtain exactly the same performance as the fixed IN, assuming that $K \geq 6$ (directed) planes.

3.1 HEURISTIC ALGORITHM

The PTCBC problem is computationally difficult, thus, in what follows we present a heuristic algorithm to solve it. We decompose the problem by first solving the partitioning problem (P) and then the topology-configuration with bounded connectivity degree (TCBC) problem. The proposed heuristic involves 3 phases that are described in the following paragraphs.

Partitioning into Logical Clusters

In the first phase we use spectral clustering to partition the traffic matrix $\boldsymbol{\Lambda}$ of size ZxZ and form logical clusters (LCs) of size T each. We obtain thus a new traffic matrix $\boldsymbol{\lambda}$ of size NxN that captures inter-LC traffic in bytes. We use the 2-way spectral clustering algorithm and apply it recursively [22] until we obtain partitions of the target cardinality (T).

Demand Ordering and Simulated Annealing

The sequential algorithm to be described next, establishes optical links based on the demands of traffic matrix $\boldsymbol{\lambda}$, by serving the demands one-by-one in some particular order. The number of optical links emanating/terminated at an optical aggregation point is constrained due to bounded connectivity. Due to this constraint, not all demands in $\boldsymbol{\lambda}$ can be served by a single optical circuit and some have to be served by multi-hop transmissions. Demands that are served earlier have higher probability of finding free resources to establish an optical link, as opposed to demands that are served later. Thus, different orderings result in different topology-configuration solutions with different costs (hop-bytes). In this work, we employ the Highest Demand First (HDF) ordering policy: we order the demands based on communication volume, and serve the demand with the highest communication volume (in bytes) first. According to this policy, heavy demands that have high communication sizes are served first. Since the performance metric of interest is hop-bytes, serving the demands with the highest volume by single-hop transmissions is a rational way to minimize the specific metric.

A number of other policies can be easily defined, based on other parameters of the traffic matrix $\boldsymbol{\lambda}$, e.g. based on the total load from a source to all destinations, the connectivity bound, etc. However, since the performance depends on many parameters, it is quite difficult to come up with a very good ordering policy. Thus, to find good orderings, we use the Simulated Annealing (SA) meta-heuristic. Specifically, we start with an HDF ordering and calculate its cost by sequentially serving the demands, using the heuristic algorithm described in the following paragraph (the cost function is the "fitness function" in the SA terminology). For a particular ordering $((s_1,d_1), \ldots, (s_k,d_k))$ of the demands, we define its neighbor as the ordering where (s_i,d_i) is interchanged with (s_j,d_j) for some i and j. Note that $k \leq N\cdot(N\text{-}1)$, depending on the number of non-zero point-to-point flows. To generate a random neighbor, we choose the pivots i and j with uniform probability among the demands. We use this neighbor generation procedure and the sequential demand heuristic as the fitness function in a typical SA iteration process.

Sequential demand serving heuristic algorithm

The algorithm described here serves sequentially the traffic between the LCs described in $\boldsymbol{\lambda}$, by configuring the OCS network to establish optical links to connect two LCs. To keep track of the configuration of the OCS network we use two integer vectors to hold the number of outgoing and incoming connections that are established for each LC. In particular, we denote by $\boldsymbol{O}= [O_i]=(O_1, O_{2,}\ldots, O_N)$, the vector of size N (N is the number of LCs) where each element O_i corresponds to the number of established outgoing optical connections from LC_i. Similarly, we denote by $\boldsymbol{I}= [I_i]=(I_1, I_{2,}\ldots, I_N)$ the vector that keeps track of the incoming connections. We also keep a set $\boldsymbol{L}$ of established optical links in the form of (i,j) LC pairs.

The demands are served sequentially in the order specified in the previous phase. For a demand, e.g. (s,d), we establish an optical link between s and d as long as the maximum connectivity degree bound is not exceeded. In particular, we check if s has free outgoing ports ($O_s \leq K$) and also if d has free incoming ports ($I_d \leq K$). If both constraints are satisfied, we establish a direct optical connection between s and d by cross-connecting the respective optical switch ports. Subsequently, we increase the O_s and I_d elements, update the set $\boldsymbol{L}$ of established links by appending (s,d) to it and continue with serving the next demand. If the connectivity constraint for s and/or d is not met, we move to the next demand. Once we have finished with all demands, the links to be established are given by the set $\boldsymbol{L}$. We use the graph formed by

LCs as vertices and $\boldsymbol{L}$ as edges as input to Johnson's algorithm and compute the all-pairs shortest paths. The set of computed shortest paths is used to calculate the average hop-bytes of the solution. To improve performance, we additionally use Simulated Annealing (SA): the sequential heuristic of this phase is run multiple times for different orderings. For each ordering, a new set of established optical links and thus set of (shortest) paths is found, yielding a new average hop-bytes value. We keep the solution that produces the lowest hop-bytes value. By controlling the number of SA iterations, we trade-off optimality for computation time.

The heuristic algorithm presented above is of polynomial complexity. The spectral heuristic used in the first phase is generally considered efficient for solving partitioning problems. The second phase involves the ordering of N^2 elements, while the third phase involves the execution of Johnson's algorithm that takes $O(|\boldsymbol{V}|^2 log|\boldsymbol{V}| + |\boldsymbol{V}||\boldsymbol{L}|)=(N^2 logN + N^2 K)$ time, which is faster than Floyd-Warshall's algorithm for sparse networks as the one considered here ($K<<N$). Last, if SA is used, the number of iterations drives the times that the third phase is executed.

4 PERFORMANCE RESULTS

We performed experiments to estimate the performance of the proposed PTCBC algorithm in a hybrid EPS/OCS IN. We considered a number of design choices of the hybrid IN, in particular different levels of optical aggregation and various numbers of optical planes. We also compare the results obtained in the hybrid IN to a torus-like electronic-only IN.

To estimate the performance of the hybrid IN, we implemented the heuristic algorithm that solves the PTCBC problem. To obtain comparison results for the electronic-only IN we used a hierarchical mapping heuristic algorithm that takes as input the logical communication graph of an application, partitions the logical entities into logical clusters (LCs) and then assigns the LCs to compute resources of a fixed IN topology to minimize the hop-bytes. The heuristic used follows the approach in [10]. It first performs a partitioning using spectral clustering, and then it assigns the created LCs to compute resources. It starts with an initial assignment and improves over that using the SA pivoting process.

To evaluate the efficacy of our approach with pragmatic traffic input, we captured the logical communication graphs of two kernels of representative HPC applications. In particular, we installed SuperLU and FFTW applications on a cluster located at IBM Dublin. SuperLU performs LU factorization of a sparse matrix and FFTW performs forward and inverse Fast Fourier transformations. We used the IPM monitoring tool [20] to capture the MPI point-to-point traffic that is generated by these applications. The applications were run on 240 up to 1920 ranks. Fig. 3 presents the point-to-point communication graphs obtained after a single run of each of the tested applications on 480 ranks. Various executions were performed to verify that the communication graphs of these applications exhibit similar patterns irrespective of the input and thus these applications fall within the static category, as described in Section 1. The captured logical communication graphs were used as input for evaluation of our approach, as well as for performing hierarchical mapping on the electronic-only IN. Given the lack of a widely-accepted method to scale these graphs, we extrapolated to higher scales by producing synthetic traffic matrices that are isomorphic to the captured ones, and to generate traffic for large problem executions that we were not able to perform on our cluster.

We assumed that the hierarchy of the system consists of 12 cores per node and 40 nodes per rack (total of 480 cores per rack), driven by the specifications of the IBM cluster. We report results for the case where the OCS network is either connected directly to compute nodes (where T=12) or to rack switches (where T=480), and for K=4,6 and 8 parallel OCS planes. For comparison purposes we also estimated the performance of electronic-only 2D-,

3D, and 4D-torus INs, (thus having the same connectivity degrees as the corresponding OCS networks). For both hybrid and electronic-only INs we used 100 SA iterations.

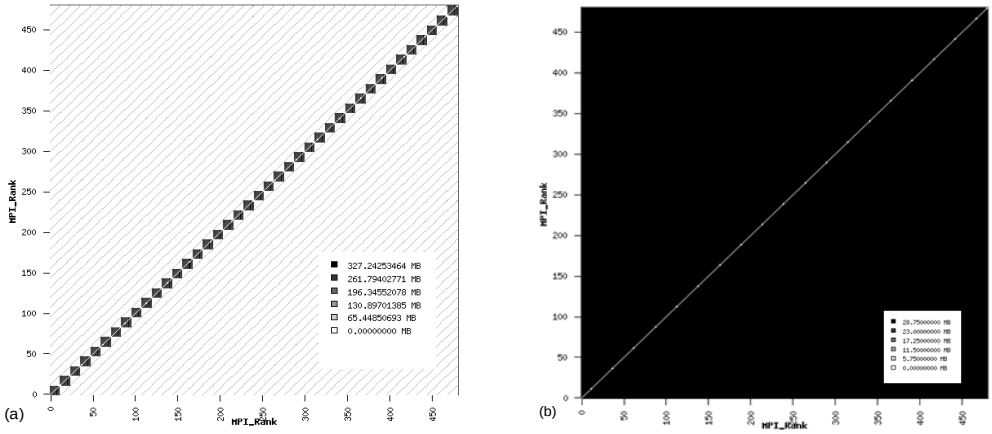

Fig. 3. Logical communication graphs of SuperLU and FFTW kernels on 480 mpi-ranks.

Tables 1 and 2 report results for SuperLU, assuming that the OCS network is connected directly to compute nodes; also, that a torus EPS network is connected directly to compute nodes. More specifically, Table 1 reports the results using the communication graphs that were captured by executing SuperLU to factorize the webbase-1M matrix taken from [23], on 240, 480, 960, 1920 MPI-ranks (N=20,40,80 and 160). Note that we used Mbytes to measure the volume of data and thus the values reported in the tables are measured in hop-Mbytes. From Table 1 we can observe that the hybrid IN exhibits lower hop-Mbytes than the electronic-only IN. Even a hybrid IN with K=4 parallel optical planes has lower hop-Mbytes for almost all problem instances examined than a 4D electronic-only IN, which has double connectivity degree. Note that the improvement that we obtain when using the hybrid IN does not come from the highest capacity supported by the OCS network. The hop-bytes metric used here does not consider the capacity of the underlying networks. Instead, the improvement comes from the configurability of the OCS part of the hybrid IN that was exploited to serve efficiently the communication graph of the application. This confirms our expectation that performing static configuration improves communication performance.

Table 2 reports results obtained when using our custom built traffic generator. In particular, we generated traffic matrices to emulate the execution of SuperLU on N=20, 40, 80, 160 and 320 nodes. Fig. 4 presents the relative hop-bytes improvement brought by our approach applied in a hybrid IN over a 3D-torus electronic-only IN. Captured traffic (indicated by webbase matrix used as input) and synthetically generated traffic results are depicted in Fig. 4. By comparing results up to 160 nodes we observe that the improvement we got using the synthetically generated traffic is in-line with the one obtained with real input. As expected, the improvement is higher as we increase the number of parallel OCS planes. A 3D torus network cannot be fully constructed on a low number of nodes (up to a few tens of nodes, e.g. 40), so the improvement obtained by the hybrid IN as opposed to the 3D torus networks for problems of that size is partially explained by the 3D torus deficiency. For K=6 (equal connectivity degree to the 3D torus), we observe that the improvement is approximately 24% when executing SuperLU on 80 nodes, and increases to 30% and 38% for 160 and 320 nodes, respectively.

Table 1: Performance for communication graphs captured by running SuperLU using the "webbase-1M" matrix as input

	Hybrid IN hop-Mbytes			Electronic-only IN hop-Mbytes		
Input	K=4	K=6	K=8	2D	3D	4D
webbase-1M (*N*=20 nodes)	5235	4046	3730	6973	5604	5604
webbase-1M (*N*=40 nodes)	12703	9944	8683	13537	13272	12724
webbase-1M (*N*=80 nodes)	16164	12622	11136	23514	16784	16481
webbase-1M (*N*=160 nodes)	37868	27381	23861	60685	41362	36965

Table 2: Performance results for synthetically generated communication graphs for the SuperLU application

	Hybrid IN hop-Mbytes			Electronic-only IN hop-Mbytes		
Size	K=4	K=6	K=8	2D	3D	4D
N=20 nodes (1/2 rack)	5221	4170	3869	7178	5594	5594
N=40 nodes (1 rack)	12915	10436	9172	14620	13438	13438
N=80 nodes (2 racks)	32289	26006	23400	62141	33142	33142
N=160 nodes (4 racks)	85586	64414	56549	148767	89936	77521
N=320 nodes (8 racks)	284455	172890	146990	392423	274562	178720

Fig. 5 presents the improvement over a 3D electronic-only IN topology for the case where the OCS network is connected to Top of Rack switches (ToR). Note that in this set of experiments we neglected the performance of the network that interconnects nodes within a rack, and we only consider inter-LC (or in this particular case inter-rack) traffic. We report results using synthetically generated traffic to emulate the execution of SuperLU from 8 up to 128 racks. Note that for small scale problem instances, the 3D torus cannot be created and the comparison results are thus unfair. However, for larger scale-out instances, the comparison indicates that the hybrid IN performs up to 35% better than the electronic-only IN.

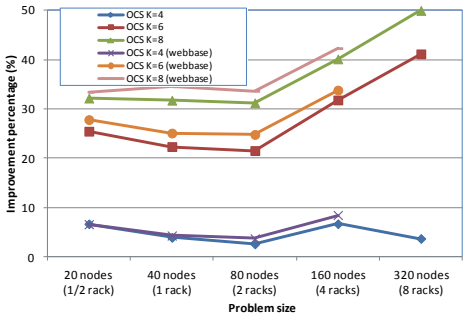

Fig. 4. Hop-bytes improvement of the hybrid IN over the 3D torus electronic-only IN (node-level aggregation).

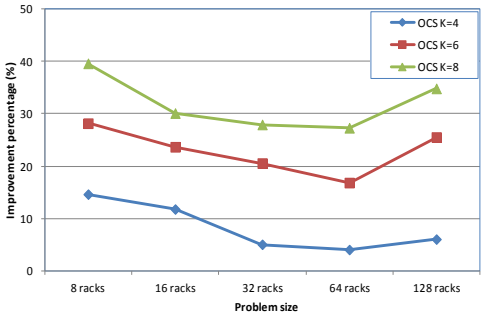

Fig. 5. Hop-bytes improvement of the hybrid IN over the 3D torus electronic-only IN (rack-level aggregation).

Similar improvement was observed for FFTW (not reported here due to space limitations).

Capital cost comparison

In addition to performance evaluation, we also created simple models to estimate the cost of the hybrid IN and compare it to alternative electronic-only solutions. We let C_{OCS} be the cost of an OCS port, and C_{EPS} and C_{TR} be the cost of the EPS port and the EPS plug-in transceiver, respectively. Assuming that optical aggregation is performed at the rack level, the cost of the electronic edge (Top-of-rack switch), the electronic part of the core network and the optical part of the hybrid IN is $C_H=N\cdot K\cdot(C_{OCS}+C_{EPS}+C_{TR})+M\cdot(C_{EPS}+C_{TR})+N\cdot R\cdot(C_{EPS}+C_{TR})$, where N is the number of racks, K is the number of parallel OCS planes, M is the number of EPS ports of the second level, and R is the number of electronic ports of edge that are connected to the compute nodes (assuming R compute nodes per rack). The cost of the electronic-only IN where the racks are connected in an X-D torus topology and the compute nodes are directly connected to the top-of-rack switches is $C_E=N\cdot 2\cdot X\cdot(C_{EPS}+C_{TR})+N\cdot R\cdot(C_{EPS}+C_{TR})$, where X is equal to 2, 3, or 4 for a 2-, 3-, or 4-D torus topology, respectively.

Due to extreme price volatility in an evolving technology domain, any specific price trend assumption would be speculative and may not survive over time. Instead, we conducted a cost comparison that is parametric to R_{OCS}. We define R_{OCS} as the relative cost of an OCS port over the cost of an EPS port plus the cost of an electro-optical transceiver, i.e. $R_{OCS}=C_{OCS}/(C_{EPS}+C_{TR})$. In this comparison we assume R=40 compute nodes per rack and M=1 port for the second level of the EPS network in the hybrid IN system.

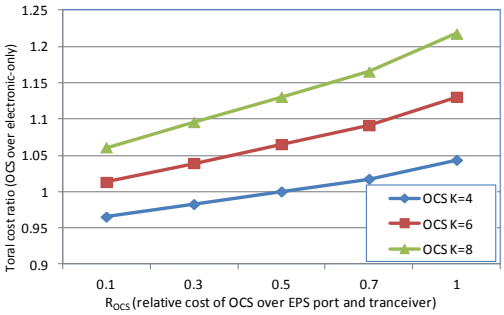

Fig. 9: Ratio of the cost of the hybrid IN over the cost of the electronic-only network that interconnects the racks in a 3D torus topology

We found R_{OCS} to be currently in the [0.5-0.6] interval, given the cost of an OCS port C_{OCS} to be around $500 [16-17] and 40Gbps EPS technology. The price of OCS switches will tend to fall rapidly, as high as 80% [24], as vendors pursue widely deploying OCS in DC, indicating R_{OCS} reduction to around 0.15 (without considering in this calculation the reduction in the cost of the electronics). A benefit brought by the proposed hybrid IN that is not factored in the cost model is that it is future-proof, since the OCS network is agnostic to protocols, modulation formats and data rates. Thus, to increase the capacity of the hybrid IN system we only need to upgrade the electronic-edge that accesses the OCS network. This also indicates that the cost of the OCS port and that of the hybrid network will almost certainly not increase, while the cost of the electronic-only IN might increase rapidly, especially if we move to a newer technology with higher bandwidth (e.g. 100Gbps). Lastly, note that the optical technology, due to its transparent nature, consumes much less energy compared to active electronic switching, a cost-saving factor not captured in this model.

5 Conclusions

We presented a method to partition the logical tasks and identify the topology configuration of the (reconfigurable part) of a hybrid EPS/OCS interconnection network (IN) to efficiently serve the point-to-point traffic produced by a parallel application with known logical communication graph. The method presented is general and can be used in many different settings, irrespective of the level at which the optical network is deployed and the number of parallel optical planes. We used the proposed algorithm to estimate the performance of the target hybrid IN and compare it against application mapping on conventional fixed, electronic-only IN that are based on toroidal topologies. Our results indicated that the hybrid IN can exhibit better average hop-bytes performance than electronic-only IN based on toroidal topologies with higher connectivity degrees. Subject to the relative costs of the electronic and optical ports, these performance improvements can come at a slightly higher but comparable

cost, while the data rate agnostic and transparent nature of optical technology ensures better upgradability and lower power consumption for the hybrid IN.

Acknowledgements
This work has been partially supported by Industrial Development Agency (IDA) Ireland and the Irish Research Council for Science, Engineering and Technology (IRCSET).

REFERENCES

1. A. Geist, "Paving the roadmap to exascale," SciDAC Review, Special Issue on Information Technology in the Next Decade, vol. 16, pp. 52–59, 2010.
2. R. Brightwell, et. al."Challenges for High-Performance Networking for Exascale Computing", Invited paper, ILCCN 2010
3. Cray Inc. Cray XT Specifications. http://www.cray.com/Products/XT/Specifications.aspx, 2009.
4. Y. Ajima, S. Sumimoto, and T. Shimizu, "Tofu: A 6d mesh/torus interconnect for exascale computers," Computer, vol. 42, pp. 36–40, 2009.
5. S. H. Bokhari, "On the Mapping Problem", IEEE Trans. Computers, 30(3):207-214, 1981.
6. B. G. Fitch, A. Rayshubskiy, M. Eleftheriou, T. Ward, M. Giampapa, M. C. Pitman. "Blue matter: Approaching the limits of concurrency for classical molecular dynamics" Supercomputing, 2006.
7. G. Bhanot, A. Gara, P. Heidelberger, E. Lawless, J. C. Sexton, R. Walkup, "Optimizing task layout on the Blue Gene/L supercomputer", IBM Journal of Research and Development, 2005.
8. A Bhatele, G R Gupta, L V Kale, I-H Chung, "Automated Mapping of Regular Communication Graphs on Mesh Interconnects", Computer Science Research and Tech Reports, 2010.
9. A. Bhatele, L. V. Kale, "Heuristic-Based Techniques for Mapping Irregular Communication Graphs to Mesh Topologies", HPLC 2011
10. I-H Chung, C-R Lee, J Zhou, Y. C. Chung, "Hierarchical Mapping for HPC Applications", International Parallel & Distributed Processing Workchop (IPDPSW), 2011.
11. M. Al-Fares, A. Loukissas, A. Vahdat, "A Scalable, Commodity Data Center Network Architecture", SIGCOMM, 2008.
12. T. Benson, A. Akella, D. A. Maltz, "Network traffic characteristics of data centers in the wild", Conference on Internet measurement (IMC), pp. 267-280, 2010.
13. A. Greenberg, et. al., "VL2: A Scalable and Flexible Data Center Network", SIGCOMM 2009.
14. K. J. Barker, et. al., "On the Feasibility of Optical Circuit Switching for High Performance Computing Systems", Supercomputing 2005.
15. L. Schares, et. al., "A Reconfigurable Interconnect Fabric with Optical Circuit Switch and Software Optimizer for Stream Computing Systems", Optical Fiber Communications (OFC), 2009.
16. N. Farrington, et. al., "Helios: a hybrid electrical/optical switch architecture for modular data centers", SIGCOMM 2010.
17. G. Wang, et. al., "c-Through: Part-time Optics in Data Centers", SIGCOMM 2010.
18. K. Asanovic, et. al., "The Landscape of Parallel Computing Research: A View from Berkeley", Technical report, Berkeley, 2006.
19. S. Kamil, L. Oliker, A. Pinar, J. Shalf, "Communication Requirements and Interconnect Optimization for High-End Scientific Applications", Transactions on Parallel and Distributed Systems, 2009.
20. Integrated Performance Monitoring (IPM): http://ipm-hpc.sourceforge.net/
21. V. Liberatore, "Circular Arrangements", International Colloquium on Automata, Languages and Programming (ICALP), 2002.
22. J. Shi, J. Malik, "Normalized Cuts and Image Segmentation", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 22, No 8, 2000.
23. http://www.cise.ufl.edu/research/sparse/matrices/
24. http://www.lightreading.com/document.asp?doc_id=213809&f_src=lightreading_gnesws