

Optimized Barriers for Heterogeneous Systems Using MPI

Jan C. Meyer and Anne C. Elster

Norwegian University of Science and Technology

Dept. of Computer and Information Science

Sem Sælands v. 7-9, NO-7491 Trondheim, Norway

{janchris,elster}@idi.ntnu.no

Abstract—The heterogeneous communication characteristics of clustered SMP systems create great potential for optimizations which favor physical locality. This paper describes a novel technique for automating such optimizations, applied to barrier operations. Portability poses a challenge when optimizing for locality, as costs are bound to variations in platform topology. This challenge is addressed through representing both platform structure and barrier algorithms as input data, and altering the algorithm based on benchmark results which can be easily obtained from a given platform. Our resulting optimization technique is empirically tested on two modern clusters, up to eight dual quad-core nodes on one, and up to ten dual hex-core nodes on another. Included test results show that the method captures performance advantages on both systems without any explicit customization, and produces specialized barriers of superior performance to a topology-neutral implementation.

Keywords—topology; adaptive; barrier; MPI;

I. INTRODUCTION

The increasing number of cores in parallel architectures introduce complexity in the interconnect, forcing designs to consider which levels of fast local memory should be private and shared, how to maintain coherency, and how to implement communication with remote systems. This presents parallel applications with the challenges of a heterogeneous infrastructure for communication. Recalling the historical development of cache-coherent NUMA machines at larger scale, a growing core count coupled with a mixture of private and shared cache memories suggests a similar tendency towards shared resources with non-uniform access times even at the chip level. Our previous work on mutual exclusion [12] with both ccNUMA

interconnects and multithreaded processors suggests that variations in signal latency caused by the physical locality of threads become significant already at modest scales, making it an important factor to control for efficient synchronization.

This paper explores a method for automatically constructing signaling patterns which form cost-efficient barrier operations, in a scenario of controlled process locality, and highly variable point-to-point signal costs. Specifically, barriers are represented as incidence matrices of layered dependency graphs, which are coupled to matrices of point-to-point signal costs in order to derive a heuristic function for overall cost. This representation allows the internals of barrier operation to be automatically specialized to the underlying topology in ways which would require a handwritten approach to rely on the topological details of the target platform. Testing the resulting, generated algorithms on clusters of dual quad-core and dual hex-core nodes show favorable performance compared to the common tree algorithm, even though that algorithm already strongly favors neighborhood locality.

The rest of the paper is structured as follows: Section II describes related work. Section III provides a high-level outline of our method, before Sections IV and V detail its topological and algorithmic aspects, respectively. Section VI shows that the combined model can predict algorithm and topology interactions well enough to guide optimization. Section VII proposes an automatic method to generate customized barrier algorithms for a profiled platform, and shows results with superior performance and scalability to the provided

barrier primitive on two systems. Finally, Section VIII summarizes our findings, and indicates directions for further research.

II. RELATED WORK

Hoefler *et. al.* [7] survey software barrier approaches, classifying them according to whether or not they include an explicit wakeup phase, and suggesting that the optimal choice for software libraries must take architectural aspects into account. Chen and Watson[8] study the impact of the memory hierarchy on SMP machine synchronization, using shared memory techniques to optimize a queued barrier algorithm on a dual quad-core server. Several works study the performance improvements attainable by adding hardware support to synchronization. Ramakrishnan and Scherson [9] study hardware augmentations to support the often overlooked aspects of simultaneous, disjoint barriers, as well as nested barriers. Villa *et. al.* [10] study the performance of barrier algorithms similar to those studied here, implementing them in terms of send and receive operations, and examine the impact of varying the on-chip topology of a simulated NoC. Sartori and Kumar [11] examine the tradeoffs of mapping tree barriers to mesh CMPs, suggesting that well-mapped virtual topologies can attain latency reductions comparable to augmenting the mesh with additional links. Their approach is similar to ours in that it leverages control of physical locality, but differs by hiding locality concerns from the software implementation through hardware support for mapping virtual topologies. The benefits in terms of flexibility and software overhead reduction attainable through such support seem very promising for future CMPs and dedicated supercomputer interconnects, but the awareness of global system structure it requires can make it challenging to provide beyond the node level of commodity clusters such as our target platforms. Vadhiyar *et. al.* [4], recognize and automatically explore topological optimizations of more general collective communications, as do Faraj and Yuan [3]. Both studies leverage empirical results in an automated tuning process, with Faraj and Yuan's study being particularly close to ours in its re-

striction of optimizations to specific topologies. Matrix representation of pairwise communication characteristics is an example of what Lastovetsky *et. al.* [1] classify as a straightforward heterogeneous extension of a traditional communication model. Our topology model is less general than their cost functions, but its simplicity is beneficial for efficiently evaluating candidate signal patterns.

Goldman *et. al* [2] study aspects of the Message Exchange Problem (MEP) in a number of scenarios, representing the problem using weighted graphs and their corresponding matrices. They explore the trade-off between minimizing the number of startup costs and the aggregate bandwidth consumption, using a hybrid algorithm which favors minimizing startup times towards the end of an exchange. Consideration of variable size messages on a uniform interconnect is equivalent to our consideration of fixed size messages on a nonuniform interconnect, in the sense that transmission time is a product of size and speed. As latency cannot be partitioned over multiple messages, this corresponds to non-preemptively scheduled message exchange, which is identified as NP-hard [2].

Although the problem of determining *optimal* message schedules may be intractable, the costs of ordinary, regular synchronization patterns differ significantly with process localities, producing potential performance gains from combining heuristically favorable algorithms. The novelty of our approach lies in treating the pattern of signals in a barrier as mutable, effectively enabling production of barrier functions which combine the different internal stages of several algorithms.

III. METHOD OVERVIEW

Our method consists of two independent models: one topological cost model which captures the performance of all pairwise communication costs in a parallel system, and one algorithmic model consisting of a sequence of boolean matrices which describe the signaling pattern of a barrier. Two clusters of multithreaded nodes are used as testbenches: one 8-node system of dual quad-core processors per node, and one 10-node system of dual hex-core processors.

The first part of the method is to collect a map of platform topology through a series of pairwise performance tests. In order to admit the same benchmarking method on the node and system levels, we rely on the OpenMPI synchronized send mechanism to send zero-length messages from source to destination, making local completion an indication that both processes have been involved in the operation. The locality of source and destination is enforced using the `sched_setaffinity` process affinity control of Linux, and a small initializer routine to provide a one-to-one mapping between MPI rank and processing core on a system-wide basis. Since process/core affinity is not yet standardized across operating systems, this technique does not produce portable programs, but it suffices for the purpose of demonstrating the benefit of affinity control. In order to verify that the resulting topological map is accurate enough to guide optimizations, three barrier algorithms are tested empirically, comparing performance to the cost implied by the topology matrices collected from our two platforms.

The second part of the method is to start with a set of barrier algorithms of known properties, and a program to predict the cost of their combinations. Once the hybrid method of minimal cost is found, the program emits the source of a library function which hard-codes the discovered pattern, and compiles it into object code. This generated function is tested and compared to the barrier operation of OpenMPI itself, showing that resulting performance is no worse, and often results in a significant cost reduction.

Figure 1 gives an overview of the two parts, illustrating their interaction. The main point is that the collection of topological profiles is decoupled from the generation and testing of particular barrier algorithms, through storing the collected maps on disk. This gives the benefits that collected profiles can be applied to estimate the cost of various candidate algorithms at low cost without occupying the target machine, as well as permitting a single program to apply the same communication patterns to produce predictions and experimental results. It also makes valid predictions require consistency

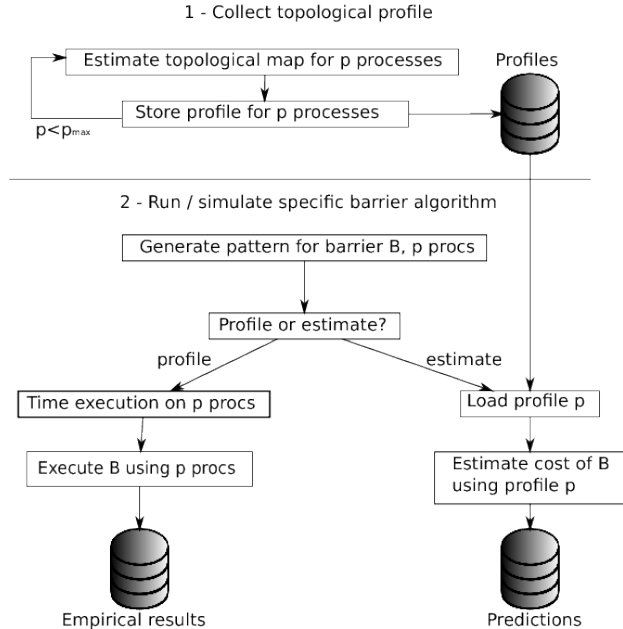


Figure 1. Overview of Model Interaction

between the run time conditions reflected in the profile and those of an experimental verification. This is managed using affinity control, as above.

The choice of target platforms and programming model is made because clusters permit experiments with greater levels of parallelism than what is found on their individual compute nodes. The hierarchical layout of the overall interconnect features a performance gap between inter-node and intra-node communication which overshadows the effects of the similar on-chip vs. off-chip and shared-cache vs. private-cache hierarchies on the single node level, thus making the dominant performance parameter a matter of locality at the node level. Still, we show that locality control at the node level exposes also the hierarchical nature of the local memory subsystems, suggesting that the larger scale effectiveness of our approach would be applicable also at finer granularity. The use of MPI is to facilitate benchmarking of a uniform signaling mechanism at both the shared-memory level on a node, and between nodes. It can be argued that this is a heavy-weight choice for the shared memory level. Our method, however, only requires maintaining processor affinity, and that a reliable cost metric can be established for point-

to-point signals. Experiments with nodes of greater local parallelism could thus employ the same basic approach, parametric in the performance figures of a more lightweight signaling mechanism.

IV. TOPOLOGICAL MODEL

In an environment with no programmatic distinction between signals traversing the various layers of the multi-layered interconnect, we approach the cost of transmitting a minimal message as a measure of topological distance, disregarding bandwidth constraints as they have negligible impact on synchronization cost. The number, variety and complexity of the software and hardware subsystems involved in the transmission of a single message rapidly make it infeasible to explicitly acknowledge them all in an exact cost function. Therefore, we simplify the costing of messages to be expressed in three empirically measurable parameters, which yield sufficient accuracy for our purpose:

- 1) The cost O_{ij} of sending *one* message between processes i, j .
- 2) The cost O_{ii} of initiating a transmission with *zero* messages
- 3) The cost L_{ij} of *adding* one message between processes i, j to a non-empty list of messages to be sent simultaneously from i

For a P -process setup, these are collected in $P \times P$ matrices O and L , with the diagonal of O estimating of the software overhead of invoking any communication call, the off-diagonal elements of O estimating the overhead of targeting a given destination, and L estimating the latency of message transmission.

A. Benchmarks for Model Parameters

To generalize these parameters with respect to an arbitrary heterogeneous interconnect, we benchmark them for each pair of processes $(i, j) \in P \times P$, where P is the set of all processes. The only assumption made regarding interconnect structure is that links are *symmetric*, i.e. that $O_{ij} = O_{ji}$, or equivalently, that the cost of a round-trip signal can be approximated by doubling the cost of a single message. This assumption is made to simplify the

adaptive implementation in Section VII, but note that extending the cost matrices to cover asymmetric links is trivial.

From this basis, the cost of a set of signals sent from a process i to a vector \vec{J} of recipients can be computed, using equations 1 and 2. Equation 1 reflects the expected total transmission time in most cases, while Equation 2 captures the case when receiving processes are known to already await signal arrival.

$$t(i, \vec{J}) = \max_{1 \leq k \leq |\vec{J}|} \{O_{ij_k}\} + \sum_{k=1}^{|\vec{J}|} L_{iJ_k} \quad (1)$$

$$t(i, \vec{J}) = O_{ii} + \sum_{k=1}^{|\vec{J}|} L_{iJ_k} \quad (2)$$

Benchmarking to find these values proceeds by a sequence of $\frac{|P|(|P|-1)}{2}$ pairwise round-trip tests to establish $O_{ij}, L_{ij} | i \neq j$, and another $|P|$ tests for O_{ii} .

The measurement used for $O_{ij} | i \neq j$ is a common cost estimate for use with the Hockney[6] model: measuring repeated transmissions of growing data sizes, fitting a linear regression line by the method of least squares, and taking its intercept as an estimate of startup cost. Our tests scale message sizes by powers of two, from 1 through 2^{20} bytes, and finds the arithmetic mean of 25 repetitions for each sample.

L_{ij} is found by transmitting a growing number of messages without payloads from i to j , up to 32. Similarly to the O_{ij} estimate, each sample point is a mean of 25 measurements, and the gradient of the regression line is taken as our statistic for the parameter value. The intuition for this quantity is that it estimates the *additional* expense of adding another message to a set which already targets one or more destinations. The cost of this is measurably different from the cost of the first message, as might be expected because of software startup cost and warmup effect. Together, these two quantities are sufficient to model the communication cost of an arbitrary signal pattern which does not follow immediately after another.

Finally, O_{ii} is found as the mean from 25 repetitions of initiating $|P|$ communication requests which ultimately do not cause any transmission. We estimate this overhead because the OpenMPI implementation used empirically showed performance characteristics similar to this figure when the receiving process is ready to receive at the time of sending, even though the ready send mode was not employed.

B. Model Accuracy

The method described above extracts statistics, obviously adding sampling noise to model parameters. The choice of parameters was made after explorative testing of various benchmarks with an emphasis on reproducible results; they form the simplest model we have found accurate enough for obtaining verifiable estimates of different barrier performances. More accurate measures could be obtained by a white-box approach to refine the set of parameters, but further refinement is omitted because there is sufficient detail to distinguish between our algorithms, and greater detail would not be verifiable given the noise level of overall barrier performance measurements.

It should be noted that the majority of the profiling runs which captured the parameters used in this study were only guaranteed exclusive access on a per-node basis, meaning that runs which did not allocate the full set of nodes were subject to interference from unrelated load on the machines. Such effects could be eliminated by obtaining exclusive access to obtain the measurements, but doing so would make the context of the obtained measurements less realistic, as exclusive access to the entire machine is not usually available in an applied scenario. Since results still proved to be reproducible under the conditions provided to ordinary users, investigating the attainable precision in ideal conditions is left for future study.

It bears mention that our sample sizes were purposely chosen to be quite small, as the $|P|^2$ tests required can absorb a significant amount of run time for large $|P|$. Although this profiling work needs only be done once for a given platform, and does not pose a great issue on our test systems, it

makes the consideration of the cost of model parameter estimation an important point with respect to general applicability.

Introducing a modest amount of *a priori* knowledge about interconnect structure can significantly reduce the work involved in profiling. Running the full set of tests can verify that the communication characteristics between one pair of processes on one pair of nodes does not differ radically from another pair on the same two nodes. Thus, a great deal of duplicate effort could be rationalized by constructing $P \times P$ matrices from replicating component submatrices, which capture local effects at each level of the interconnect. Our experiments do not exploit this potential reduction, as we wish to retain the benchmarking program as a constant across platforms of differing topology. This was done to ensure that no unforeseen effect would be overlooked by the assumption that all nodes behave similarly, but results did show similar submatrices corresponding to similar subsystems, suggesting that this could have been assumed and exploited without significant loss of information.

V. ALGORITHMIC MODEL

The following Sections describe the general representation of a barrier algorithm as a dependency graph encoded in a sequence of boolean matrices, and exemplifies it by showing three specific barrier algorithms represented in this manner.

A. General Representation

A barrier algorithm can effectively be represented by a dependency graph. The property that no participant may leave the barrier before all have entered, implies that this graph must contain a path from each arrival vertex to each departure vertex, but as long as this property is fulfilled, the structure of these paths is arbitrary. In order to sort the dependencies fulfilled by each act of communication for separate analysis, we choose to represent an overall algorithm as a sequence of steps $0, 1, \dots, k$, in which each process may signal any combination of other processes, where the signals sent in each step must be received before subsequent steps can begin. The layers thus induced by each step can

conveniently be represented as boolean incidence matrices S_1, S_2, \dots, S_k , with the row vector corresponding to each rank representing which targets it will signal in a given step. The convenience of this encoding is that the per-step cost of a signal pattern maps naturally onto the cost model from Section IV, permitting an automatic inference of overall cost independent of the specific communication pattern required by a given algorithm.

Whether a given pattern implies global synchronization can be tested by a sum of matrix products. Starting with the identity matrix, we have represented that each process knows of its own arrival. The distribution of this knowledge after one barrier step can be found in the matrix $K_0 = I + S_0$. Accumulating the effect of each successive step inductively gives us Equation 3. The signal pattern encoded in the sequence S_0, S_1, \dots, S_k represents a barrier if and only if all elements of K_k are non-zero.

$$K_a = K_{a-1} + K_{a-1} \cdot S_a | 0 \leq a \leq k \quad (3)$$

B. Three Barrier Algorithms

The *linear*, *dissemination* and binary *tree* barrier algorithms are used as building blocks in the following Sections. These algorithms are selected because they span a range of design choices. The linear barrier is chosen for its simplicity, the tree barrier is chosen because it is a very widely used hierarchical method, and the dissemination barrier is chosen because it is neutral to the number of participants, while belonging to the class of methods which omit explicit departure signals [7]. The *linear* barrier uses a master rank to count arrivals, and signal departure to every rank when the count is complete. In matrix form (using $|P| = 4$ for illustration purposes), this becomes the 2-stage sequence in Figure 2. The *dissemination* barrier proceeds in $\lceil \log_2 P \rceil$ stages. For each stage s , each participant i signals $j = (i + 2^s) \bmod P$, as shown in Figure 3. Finally, the *tree* barrier embodies the familiar textbook algorithm which proceeds by collecting and dispatching signals in a binary tree pattern of $2 \cdot \lceil \log_2 P \rceil$ stages, as shown in Figure 4.

$$S_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, S_1 = S_0^T$$

Figure 2. Linear Barrier in Matrix Form

$$S_0 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}, S_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Figure 3. Dissemination Barrier in Matrix Form

Both the linear and tree algorithms provide clear illustrations of a general principle which we will use, i.e. that *hierarchical* algorithms use one process as a temporary root for others to collect their arrival signals, and distribute the departure signals by means of the same matrices transposed, applied in reverse order. Using this symmetry is obviously not the *only* way, and asymmetric topologies may even profit from utilizing different patterns for arrival and departure, but we still note that this structure can produce local barrier patterns as parts of an overarching algorithm which addresses the full set of processes.

VI. MODEL VALIDATION

To validate the coupling of our two models, we test them on the two cluster platforms. One is our departmental cluster, consisting of 8 nodes with dual 2GHz Intel Xeon E5405 quad-core processors, the other is NTNUs new central cluster, where we tested on 10 nodes featuring dual 2.4GHz

$$S_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, S_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$S_2 = S_1^T, S_3 = S_0^T$$

Figure 4. Tree Barrier in Matrix Form

AMD Opteron 2431 hex-core processors. Nodes on both systems are interconnected with gigabit ethernet. The performance variability between layers of these interconnects is sufficient to illustrate the value of customizing an algorithm to the specific topology.

The program used to validate the model employs a general simulator for matrix encodings of barriers, storing the tested barrier in a structure with a stage count, as well as the sequence of incidence matrices, and an array of MPI requests to match the signal pattern of each stage. Execution amounts to each participating process looping over the required number of stages, issuing nonblocking, synchronized signals according to the dependencies of the stage (with `MPI_Issend`), and awaiting completion of all issued requests. To verify the correctness of this program, each algorithm was tested P times for each problem size, with each of the P participants introducing a 1-second delay before calling the barrier. Observing the expected delay in the execution time at every process verifies that all processes are actually synchronized.

Predictions were collected by carrying out the sequence of matrix multiplications indicated by Equation 3, weighting the incidence matrices by the cost implied by Equations 1, 2, to obtain matrices of per-rank cost estimates at each step. To obtain a reasonable performance metric for overall barrier behavior from these per-rank cost estimates, the predicted value is extracted from traversing the dependency graph from all arrivals through all departures, and reporting critical path cost.

A. Results and Discussion

Figures 5 and 6 contrast the results we get from evaluating each barrier with respect to the collected topological profile, with results obtained by empirical performance measurements. These two figures show the various algorithms' relations to each other, with algorithms being marked by their initial letters "D", "T" and "L". Figures 7 and 8 superposition the measured and predicted results, showing the results by individual algorithms.

The principal conclusion to be drawn from Figures 5 and 6 is that the combined model clearly

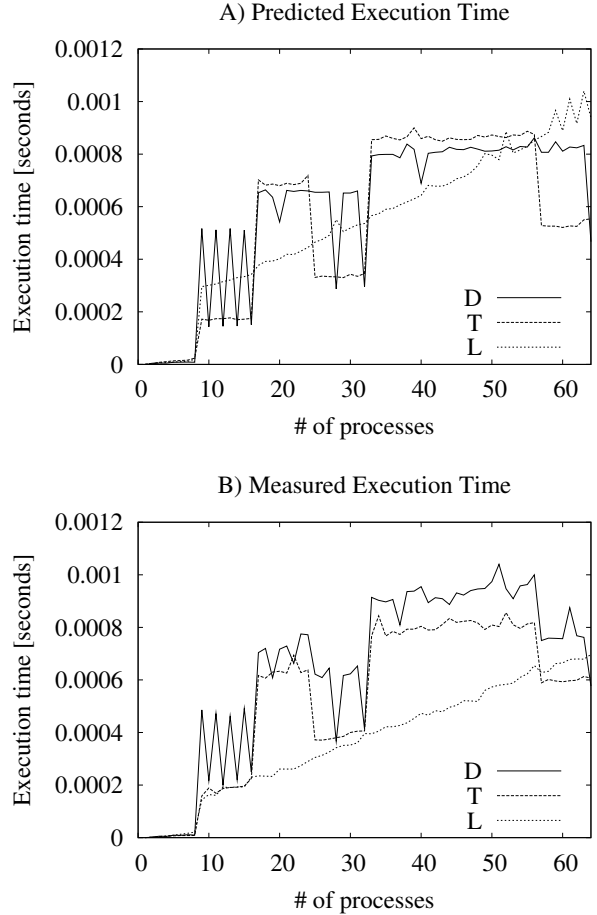


Figure 5. Validation on 8 Nodes of Dual Quad-cores

captures the interaction between the algorithm and topology. This is immediately visible from the shape of the graphs, and their relative displacements, to an error of approximately $200\mu s$; beyond this, our measurements no longer permit us to distinguish between algorithms. The error is most visible in the misprediction of the intercepts between the linear barrier and the others, but as its magnitude does not increase with scale, it represents a decreasing percentile of overall barrier execution time. Targeting systems with lightweight operating systems or dedicated low-latency interconnects at the node level may make it possible to improve the absolute accuracy of predictions. In a commodity component cluster environment, improving the absolute accuracy would likely require us to augment the cost model with terms for further

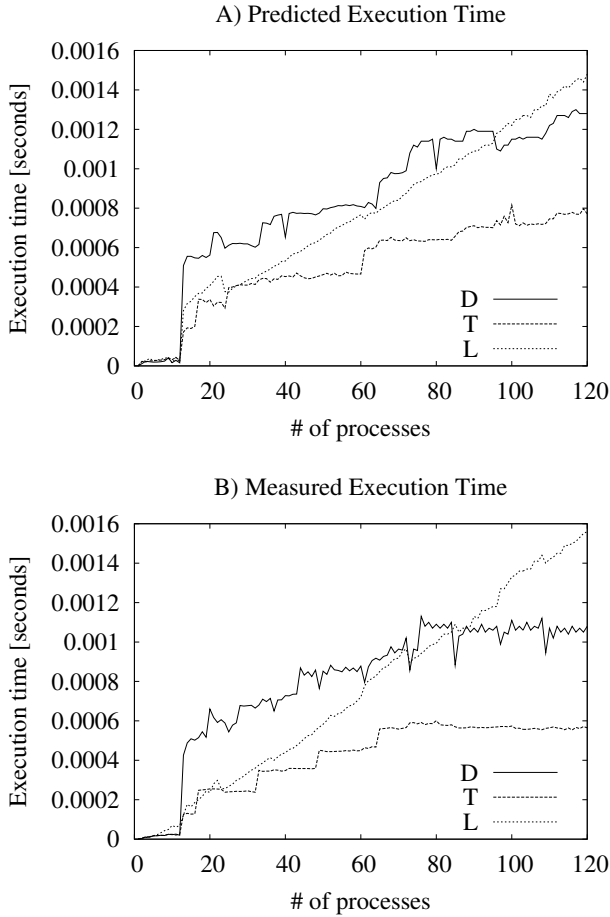


Figure 6. Validation on 10 Nodes of Dual Hex-cores

phenomena. Since our approach can work in terms of an approximate, relative cost measure, we will not address this issue further.

One point of note is that the dissemination algorithm favors problem sizes which are powers of 2, by construction. This is clear from both predicted and empirical results: on the platform with a power-of-two node size (dual quad-core), the later phases of this algorithm become resolvable by node-local signal exchanges only, and the model captures this effect almost to perfection, most visibly in the 32 and 64 process cases which fit on the cluster. Other regions of particular interest are the relative performances of the dissemination and tree barriers for the 4-node (25 through 32 process) and 8-node (57 through 64 process) cases, where the tree barrier makes reduced use of the slower

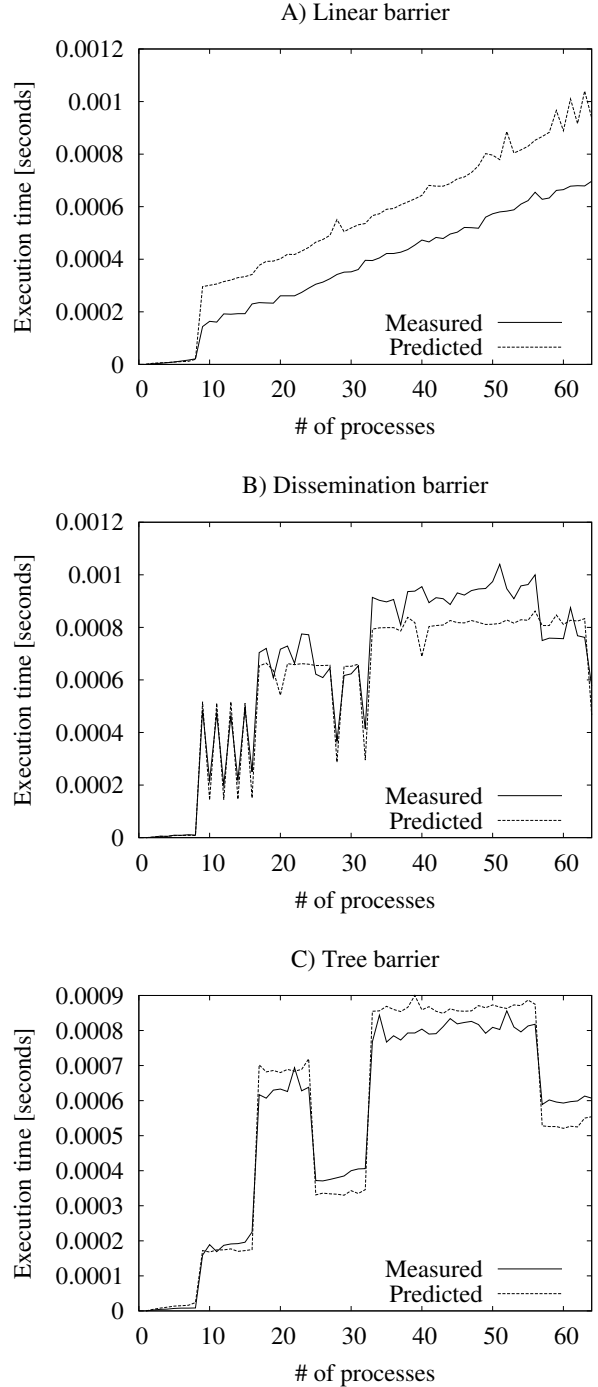


Figure 7. Individual Barriers on 8 Nodes of Dual Quad-cores

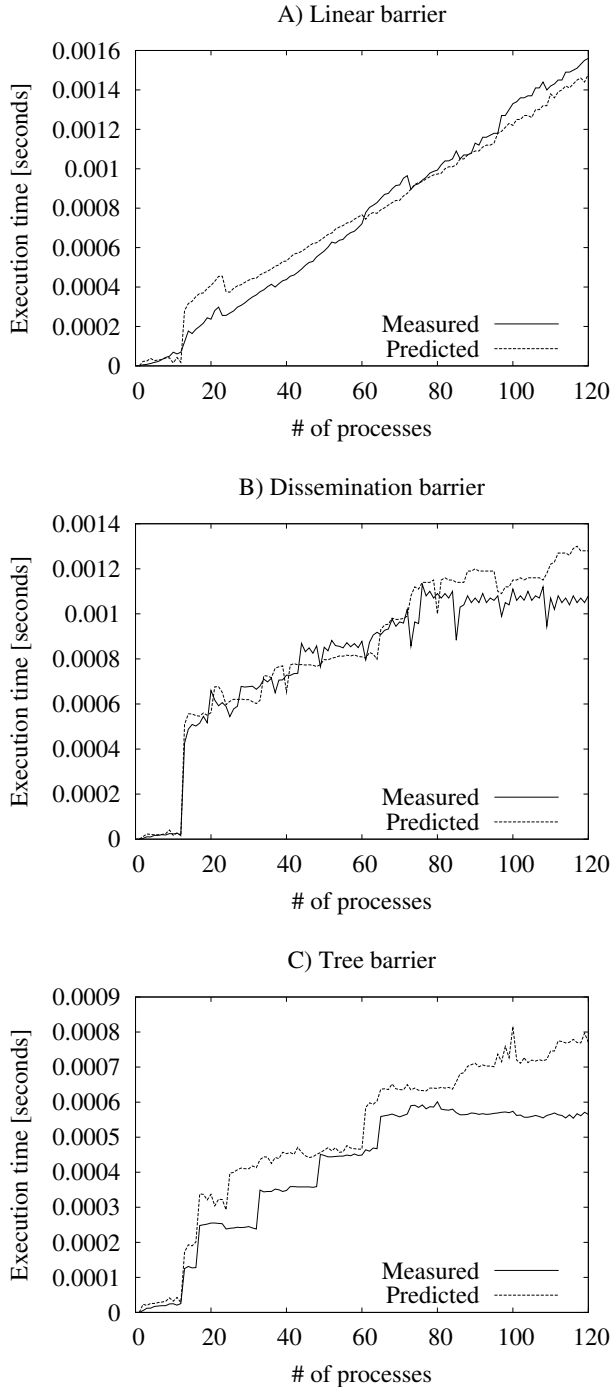


Figure 8. Individual Barriers on 10 Nodes of Dual Hex-cores

links relative to the dissemination barrier. Finally, note the oscillating effect between even and odd process counts in the 2-node (9 through 16 process) case. This effect occurs because the scheduling software on this cluster maps processes to nodes in a round-robin fashion, causing the dissemination barrier to feature phases consisting solely of traffic across slower links when its power-of-two offsets are modulated over odd values of P . As this is an effect of how software *maps onto* the hardware topology, it could obviously be eliminated, but for the purpose of these tests, the fact that the model predicts this behavior provides evidence of its ability to capture the features which affect algorithm performance at run time. The cluster of dual hex-core nodes features fewer such noticeable artifacts, as its multiple-of-12-core shared memory configuration does not coincide with special cases of the algorithms' design in the same manner as multiples of 8. The main observation to draw from these results is that the degree of model detail describes exceptional conditions without need for any particular customization.

VII. ADAPTIVE TUNING

Coupling of the algorithmic and topological models on a given platform attains sufficient accuracy to distinguish between algorithms. This allows automatic performance tuning, resulting in methods of competitive performance.

A. Rank Clustering

A common, important observation to construct optimized communication patterns is to that the layers of the interconnect divides processes into closely coupled subsets, separated by remote links which are orders of magnitude slower than local communication. While this is not true for an arbitrary interconnect, we will assume this structure for the purpose of this discussion. Developing methods for topologies which are not hierarchical will require reasoning along the same lines, but with different techniques to exploit the locality information from the profile.

A frequently used way to discover clustered subsets is by the *k-means* algorithm, which randomly selects a set of k cluster centroids, and

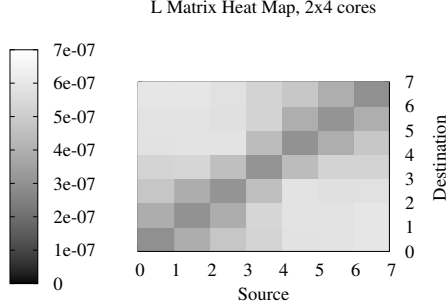


Figure 9. L Matrix Structure of One Dual Quad-core Node

iteratively updates them by modifying memberships in k corresponding clusters, until each cluster minimizes mean distance from the centroid. In principle, this would suit our purpose, but k -means unfortunately relies on a Cartesian distance metric, meaning that our topological model would have to be projected into a suitable coordinate space prior to analysis. For simplicity, we find that *sparse spatial center* (SSS) clustering [5] is adequate. This method only requires that clustered points reside in a *metric* space, i.e. non-zero distances separate non-identical pairs symmetrically, and the triangle inequality holds. The use of this method is our reason for requiring symmetry of the topological profile. With rank 0 as a member of the first cluster, and using a sparseness parameter of 35% of diameter to cluster the remaining ranks, we get clusters of node-level granularity on our test systems.

The outcome of the clustering process is a representation of the topology as a tree, with more closely connected clusters towards the leaves. The topology of our test systems result in a two-level hierarchy, but the tree construction works with any number of levels. Further lowering the sparseness parameter can refine the clustering to cores on a chip and cores sharing cache, but our cost function predictions and empirical observations already differ by 100s of microseconds, suggesting that differences at lower levels will be unobservable in

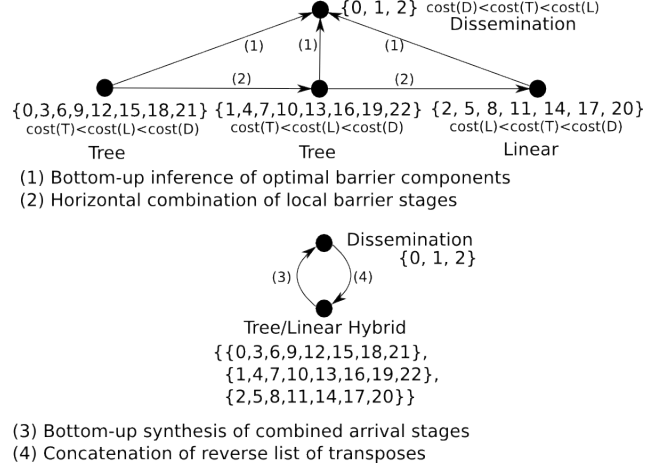


Figure 10. Construction of a Hierarchical, Customized Barrier

overall barrier performance.

Although it isn't effectively utilized in these experiments, we note that the benchmarks reveal hierarchical locality effects single-node level also, as shown in Figure 9. This figure shows the L matrix of a single-node coded by shades of grey. Note the two darker 4×4 areas encompassing ranks $[0, 3]$ and $[4, 7]$, respectively. These were verified to match operating system core mappings, and show around a factor 4 observable difference between on-chip and off-chip messages.

B. Barrier Composition

As the cost model carries a small penalty for the execution of an empty barrier stage, and three algorithms have already been tested, it is possible to find a loose upper bound on the number of stages in an optimal algorithm, and potentially search the entire space of admissible matrix sequences for the best solution. Even though it may be feasible, however, this approach is quite computationally demanding, and would examine a large range of algorithms which are quite obviously far from optimal. As our objective is to illustrate the value of the method, rather than to devise a theoretically optimal approach for our test systems, we instead take a constructive approach which tests combinations of the three basic algorithms.

The overall approach is to traverse the tree of clusters and evaluate all three algorithms on the

cluster level, greedily selecting the one with the lowest predicted cost of its arrival phases. The next step is to traverse the tree bottom-up, combining the local barriers on the same level into an overall structure for complete arrival, before inferring the departure phases by a reversed sequence of transpose matrices. Figure 10 illustrates this process for a 3-node/22-process case on 3 nodes of our dual quad-core test cluster (with round-robin mapping).

The combination of local barriers at each level of the tree produces a minor design choice, as variability in both cluster size and selected algorithm can produce partial patterns of differing numbers of stages. As an example, stage 2 in Figure 10 would require two 3-stage local tree arrival phases to be embedded in the same sequence of matrices as a 1-stage linear arrival phase. Our test program resolved this by merging shorter sequences with longer ones as early as possible, i.e. embedding the concurrent signals from ranks 5, 8, 11, 14, 17 and 20 to process 2 in the first stage of the 3-stage result, leaving the latter two stages with zeros for these ranks. This produces a single sequence of $P \times P$ patterns, which describe execution for the entire system.

The hierarchical construction with its corresponding transpositions created one exception in the cost approximation for the local barriers, since the dissemination barrier only requires a departure phase when it is used on a lower level of the hierarchy. Therefore, costs in the tree structure are approximated using the cost of each algorithm’s arrival phases multiplied by 2, except for the case where the root level is a dissemination barrier, in which case the multiplier is 1, and the subsequent matrix transpositions omit the root level. The result is that the generated hybrid algorithms favor applying the dissemination barrier to top-level uniform collections of high-latency links. This is as expected, because of the manner in which it concurrently exploits a wide range of links, to the advantage of reducing the number of communication steps.

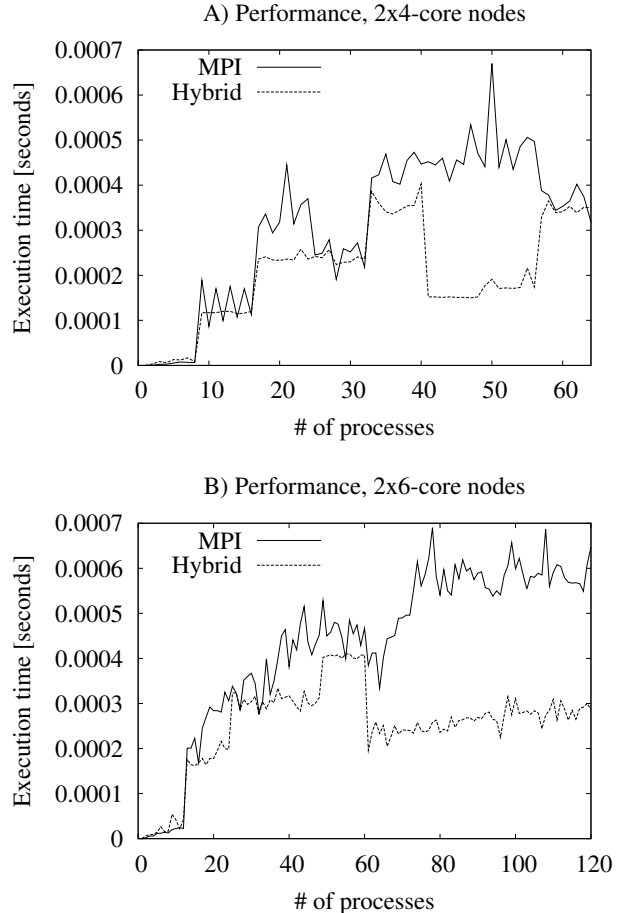


Figure 11. Performance of Generated Codes

C. Performance Results

Since their matrix representation is the working format of the adaptive construction, output is suited to execution with the general program used in the validation experiments. That would not truly test the best efficiency attainable, as the generality of treating barriers as data induces some overhead. Instead, we measure the performance of the optimized barrier algorithms after the use of a code generator, which takes a matrix sequence as input, and emits a specific barrier implemented by a hard-coded sequence of synchronous point-to-point sends.

The resulting performance figures for both our test clusters are shown in Figure 11, which include results for the hybrid and MPI barriers. The MPI measurements were obtained using the

MPI_Barrier operation of OpenMPI, which was the default choice installed on both test systems.

It is worth mentioning that as the generated test programs specialize the logic of the general model, eliminate no-op transmission steps, etc., modeling their execution would be subject to corresponding modifications to the performance model. Since the purpose of these experiments is to illustrate the models' potential for guiding the synthesis of efficient algorithms, we omit the details of adjusting the predictions for absolute accuracy. Even so, we note that features of the binary tree barrier predictions can be seen in the performance figures of the MPI barrier. This is most visible when comparing the dual quad-core node results in Figure 11 with the predictions in Figure 5; performance shows marked improvements on 4 and 8 nodes, and displays reasonably level performance for the intermediate cases. The publicly available OpenMPI library source code verifies that it implements a tree barrier.

Figure 11 shows that the adaptive barrier construction is effective. Generated barrier performance is similar to the MPI barrier at worst, and significantly improved in most cases. Notice in particular the change which occurs at the introduction of the 5th node, i.e. 40 processes on the dual quad-core configuration, and 60 on the dual hex-core configuration. At this point, execution time for the hybrid barrier drops, which indicates that a change of top-level algorithms was found profitable for these sizes. On the bigger system, this benefit halves the barrier overhead for our largest cases, which is evidence that there is a large, untapped performance potential in synchronization. Informing algorithm designs with topological information could improve both the application performance and scalability of these systems.

VIII. CONCLUSIONS AND FUTURE WORK

A model to couple detailed descriptions of barrier algorithms with profiles of heterogeneous cluster topologies has been developed, and an MPI based implementation has been validated empirically. We have also suggested a greedy algorithm to compose synchronization algorithms from smaller

component cases, and shown how it yields results of competitive performance and scalability.

Our guided optimization only searches a selected fraction of the admissible solution space for algorithms. A natural extension to this work would be to generalize it both with respect to algorithms employed as components, and constructs to broaden the space of target platforms beyond hierarchical clusters. Another appealing direction would be to employ this method in a library implementation which would benefit unmodified application codes. There are several technical impediments to this, foremost of which is the lack of a standardized programming interface for process affinity. A portable library cannot employ the strategy detailed here, because it would rely on the application to guarantee a process layout consistent with the one used to measure the topological information. Implementing a solution which stores the profile in a manner which can be efficiently indexed at run-time would alleviate this problem.

As the presented work captures its topological model statically, predictions do not consider run-time effects of contention and congestion which could be caused by background load. With a topological model ready, the generation and evaluation of adapted patterns requires on the order of 0.1 seconds, making it feasible to periodically re-evaluate the efficiency of synchronization through changing conditions. While the instrumentation required to capture incremental cost updates at run time would be relatively inexpensive, translating this into a usable scheme to dynamically alter the synchronization strategy is challenging, because the cost of restructuring the barrier would include some communication of local observations. This would only make it worthwhile to adapt the algorithm when the overhead could be amortized over a sufficient number of subsequent synchronizations. Developing an efficient scheme to estimate the profitability of dynamically altering methods makes an interesting topic for further study.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Magnus Lie Hetland, who provided useful insight on clustering

algorithms, as well as the department of computer science and the Norwegian University of Science and Technology, for providing access to the computational resources. We are also grateful to our anonymous reviewers, whose insightful comments helped improve this paper.

REFERENCES

- [1] A. Lastovetsky, V. Rychkov and M. O’Flynn, *Accurate Heterogeneous Communication Models and a Software Tool for Their Estimation*, Intl. Journal of High Performance Computing Applications Online-First, doi:10.1177/1094342009359012
- [2] A. Goldman, J. G. Peters and D. Trystram: *Exchanging Messages of different sizes*, Journal of Parallel and Distributed Computing, Vol. 66, No. 1, 2006.
- [3] A. Faraj and X. Yuan, *Automatic generation and tuning of MPI collective communication routines*, Proceedings of the 19th annual intl. conference on Supercomputing, pp. 393–402, 2005.
- [4] S. S. Vadhiyar, Fagg, G. E. Fagg and J. Dongarra: *Automatically Tuned Collective Communications*, Proceedings of the IEEE/ACM SC2000 Conference, 2000.
- [5] N. Brisaboa, O. Pedreira, D. Seco, R. Solar, and R. Uribe, *Clustering-Based Similarity Search in Metric Spaces with Sparse Spatial Centers*, Proceedings of the 34th Conference on Current Trends in Theory and Practice of Computer Science, LNCS No. 4910, pp 186–197, 2008.
- [6] R. W. Hockney, *The Communication Challenge for MPP: Intel Paragon and Meiko CS-2*, Parallel Computing, Vol. 20, No. 3, 1994.
- [7] T. Hoefler, T. Mehlan, F. Mietke, and W. Rehm, *A Survey of Barrier Algorithms for Coarse Grained Supercomputers*, Chemnitzer Informatik Berichte, 2004.
- [8] J. Chen and W. Watson: *Software Barrier Performance on Dual Quad-Core Opteron*s, Proceedings of the 2008 International Conference on Networking, Architecture, and Storage, 2008.
- [9] V. Ramakrishnan and I. D. Scherson, *Efficient Techniques for Nested and Disjoint Barrier Synchronization*, Journal of Parallel and Distributed Computing, Vol. 58, No. 2, 1999.
- [10] O. Villa, G. Palermo and C. Silvano, *Efficiency and Scalability of Barrier Synchronization on NoC Based Many-core Architectures*, Proceedings of the 2008 international conference on Compilers, architectures and synthesis for embedded systems, 2008.
- [11] J. Sartori and R. Kumar, *Low-overhead High-speed Multi-core Barrier Synchronization*, Proceedings of the 5th International Conference on High-Performance Embedded Architectures and Compilers, (HiPEAC’10), 2010.
- [12] J. C. Meyer and A. C. Elster, *Latency Impact on Spin-Lock Algorithms for Modern Shared Memory Multiprocessors*, Scalable Computing: Practice and Experience, Vol. 9, No. 3, 2008.

Jan C. Meyer (IEEE Member 2010) is a University Lecturer and Ph. D. candidate at the Dept. of Computer & Information Science, Norwegian University of Science & Technology (NTNU).

Mr. Meyer was born in Trondheim, Norway. Upon completing secondary education he was accepted to NTNU, and received an M.Sc. in computer science in 2004, which included a year of studies focusing on complexity theory at Tartu Ülikool in Tartu, Estonia. He has worked on performance tuning of scientific applications and systems administration for the Norwegian Metacenter for Computational Science (NOTUR).

Currently, he teaches courses on parallel computations and compiler construction, while finalizing his Ph.D. work at NTNU. His research interests include programming and performance models for high performance computing applications.

Dr. Anne C. Elster (Senior Member IEEE 2000) is an Assoc. Prof. of Computer & Info. Science (IDI) at the Norwegian Univ. of Science & Technology (NTNU) in Trondheim, where she founded HPC-Lab, has supervised 40+ graduate students and hosted several visitors. Her research focus is Heterogeneous and Parallel Computing, including GPU computing. She recently served as one of four IDI Section Heads, and is one of four Working Group leaders of EU COST Action IC0805: Open

European Network for High Performance Computing on Complex Environments. Dr. Elster was born near the Arctic Circle in Mo i Rana, Norway. After completing her secondary education in Porsgrunn, Norway, she received a one-year scholarship to the Univ. of Oregon. Curious to find out how computers really work, she then transferred to the Univ. of Massachusetts at Amherst where she received a B.S. in Computer Systems Engineering cum laude. While there, she also took several courses in computer science and honors mathematics. She also holds M.S. and Ph.D. degrees in EE from Cornell University where she had a lot of fun at their supercomputer center exploring various HPC systems in the late 80s and early 90s.

She worked for Schlumberger in Austin 1994-97, then part-time at Univ. of Texas at Austin in 1997-2000 and founded ACENOR Inc. in 2000 and joined NTNU in 2001.

She served on the MPI standards committees (MPI and MPI-2), on the Research Council of Norway's HPC committee (TRP III, 2003-2004) and on the Danish Research Council's Expert Panel for Research Infrastructure (2007-2010). She is currently spending her sabbatical for 2010/11 at the ECE Dept. at the University of Texas at Austin. Dr. Elster has recently given several invited talks incl. at IMA Workshop: High Performance Computing and Emerging Architectures , Jan. 2011 in Minnesota.