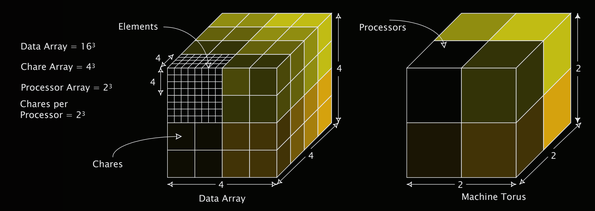
TOPOLOGY API

Task Mapping for Communication Optimization

Motivation

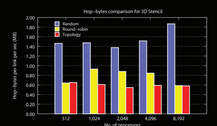
A simple 3-dimensional (3D) Stencil computation can demonstrate that topology aware mapping of objects in a parallel program leads to overall performance improvements. We did a Charm++ implementation of a 7-point 3D Stencil computation and ran it on Blue Gene/L.



The 3D data array was divided into cubes and each cube was assigned to one Charm++ Virtual Processor (VP). The runtime does a default placement of the VPs. Round-robin and topology-aware mapping schemes were compared with the default scheme and the results are presented below:

Table I. Performance (time for 100 iterations in secs) on Watson BG/L

	VP1: 1 VP per PE			VP8: 8 VPs per PE		
Cores	RND	RR	TO	RND	RR	TO
512	407.25	156.09	153.85	416.26	147.18	146.61
1024	270.56	96.48	82.19	294.59	101.82	76.80
2048	181.45	49.74	42.11	161.94	46.43	40.24
4096	115.56	25.61	21.27	93.27	24.89	23.65
8192	53.44	10.67	10.81	32.59	11.41	11.34



The round-robin scheme which is implicitly topology aware and beneficial for the near-neighbor communication of 3D stencil shows performance improvements by a factor of 5 over the default scheme. This is also reflected in the improvements in hop-bytes, which is the sum of the distances traveled by each message multiplied by their respective sizes. This suggests that the cause of message delays is congestion or specifically contention for the same links by different messages.

The topology aware scheme does even better than the round-robin scheme in most cases. This suggests that topology aware mapping is very important for good performance, especially for applications where the communication is not just near-neighbor!

Topology Manager: An API

The Topology Manager API is an interface we have developed between the application and the parallel machine. This API provides information to the application which is necessary for topology aware task placement at runtime.

Different functions provided by the API can be grouped into the following categories:

1. Size and properties of the allocated partition: Used to obtain the dimensions of the allocated partition, number of cores per node and other information such as if there are wraparound (torus) links in each dimension.

2. Properties of an individual node: Used to obtain the physical co-ordinates corresponding to a particular rank and vice-versa.

3. Additional Functionality: Mapping algorithms often need to calculate the number of hops (links) between two ranks or pick the closest rank to a given rank. The API provides such utility and other functions which are useful for mapping.

Currently this API is useful on supercomputers with 3D torus and mesh interconnects (such as Cray XT and IBM Blue Gene machines).

Topology Information on XT and BG machines

Cray XT machines: Obtaining topology information on Cray machines is a two step process:

1. Get the node IDs (nid) for all MPI ranks (pid) through the system calls cnos_get_nidpid_map and PMI_Portals_get_nidpid_map on XT3 and XT4/5 respectively. Node ID is a unique ID for each physical node.

2. The second step is obtaining the physical coordinates corresponding to a node ID using the system call rca_get_meshcoord from “rca_lib.h”.

Once we have a mapping of each rank to physical coordinates, the API calculates information such as the extent of the allocated partition (3D shape assumed).

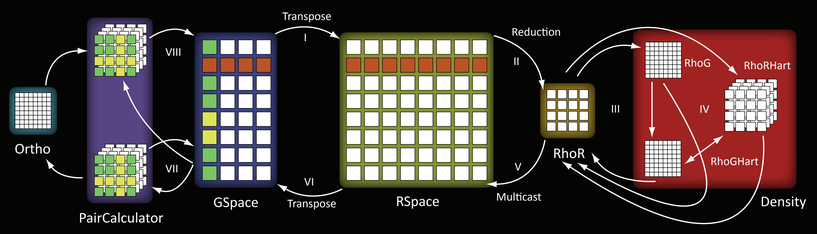
Blue Gene machines: On Blue Gene/L and Blue Gene/P, this information is available through system calls to the BGLPersonality and BGPPersonality data structures respectively. The API makes these calls and stores the information so that the application does not have to make costly system calls again and again.

The availability of this API on 3D torus and mesh machines provides a uniform interface to mapping algorithms and hence the application does not need to know if it is running on a Blue Gene or a XT machine.

OpenAtom: A Case Study

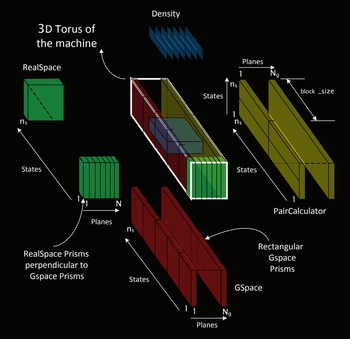
OpenAtom is a Charm++ application which does quantum chemistry computations. The application is heavily communication-bound and involves several overlapping phases with diverse communication patterns (figure below).

Important communication patterns:
- GSpace, g(state, plane) communicates with PairCalculator, p(state, state, plane) plane-wise
- GSpace, g(state, plane) communicates with RealSpace, r(state, plane) state-wise
- Optimal placement for one pattern hurts the other



A hybrid mapping which tries to optimize both communication patterns is required.

Mapping Scheme:
- The 3D partition is divided along the longest dimension into prisms
- All states of one plane of GSpace are assigned to processors in one prism
- RealSpace objects are mapped close to GSpace objects. All planes of one state of RealSpace use the same processors as the corresponding state in GSpace
- All objects in a plane of PairCalculator are placed on the same processors as the corresponding plane of GSpace
- Density and Ortho objects are placed in close proximity to the respective RealSpace and PairCalculator objects respectively.



Performance Results

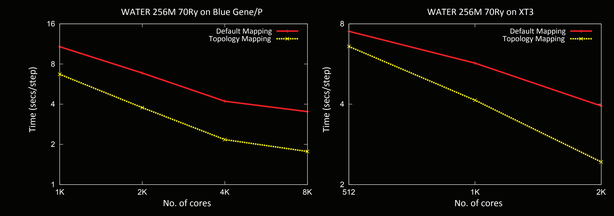
Topology aware mapping leads to nearly 40% improvement on different machines: Watson Blue Gene/L, PSC’s XT3 and ANL’s Blue Gene/P.

Table II. Performance (time per step in secs) on Watson BG/L (CO mode)

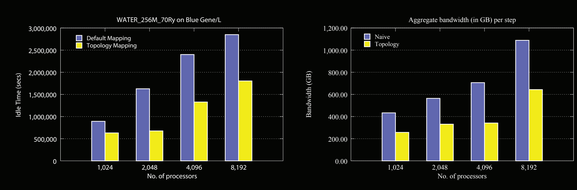
	WATER 32M 70Ry		WATER 256M 70Ry		GST BIG	
Cores	Default	Topology	Default	Topology	Default	Topology
512	0.274	0.259	-	-	-	-
1024	0.189	0.150	19.10	16.4	10.12	8.83
2048	0.219	0.112	13.88	8.14	7.14	6.18
4096	0.167	0.082	9.13	4.83	5.38	3.35
8192	0.129	0.063	4.83	2.75	3.13	1.89
16384	-	-	3.40	1.71	1.82	1.20

Table III. Performance (time per step in secs) on PSC’s XT3 (Bigben)

	WATER 32M 70Ry		WATER 256M 70Ry		GST BIG	
Cores	Default	Topology	Default	Topology	Default	Topology
			Single core per node			
512	0.124	0.123	5.90	5.37	4.82	3.86
1024	0.095	0.078	4.08	3.24	2.49	2.02
			Two cores per node			
256	0.226	0.196	-	-	-	-
512	0.179	0.161	7.50	6.58	6.28	5.06
1024	0.144	0.114	5.70	4.14	3.51	2.76
2048	0.135	0.095	3.94	2.43	2.90	2.31

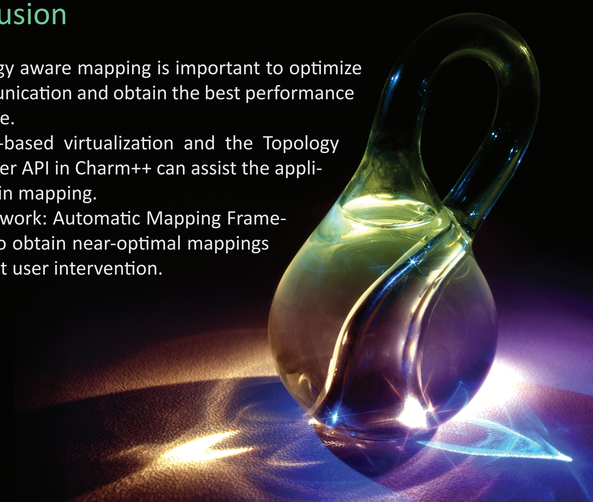


Reduction in idle time (recorded through Projections) using a better mapping illustrates that processors spend less time waiting for messages. Reduction in congestion over the network is demonstrated by the aggregate bandwidth used, as reported by IBM’s High Performance Monitor (HPM) library.



Conclusion

- Topology aware mapping is important to optimize communication and obtain the best performance possible.
- Object-based virtualization and the Topology Manager API in Charm++ can assist the application in mapping.
- Future work: Automatic Mapping Framework to obtain near-optimal mappings without user intervention.



Abhinav Bhatelé, Eric Bohm, and Laxmikant V. Kalé

PARALLEL PROGRAMMING LAB
DEPT. OF COMPUTER SCIENCE, UNIVERSITY OF ILLINOIS
PPL UIUC

Details: http://charm.cs.uiuc.edu/~bhatele/phd
Further reading: Bhatele and Kale, LSPP ’08; LSPP ’09.
Bhatele and Kale, Parallel Processing Letters, 18(4):549-566, 2008.
Klein bottle photograph and Background design by David Michael Kunzman
Acknowledgements: Staff at PSC, ANL and Teragrid, IBM Research
Funded by: CSAR DOE B341494, ORNL DOE DE-FG05-080R23332, NSF ITR 0121357