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 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/L 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_Ports_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_nodecoord 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 BGPPersonality 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

Performance Results
Topology aware mapping leads to nearly 40% improvement on different machines; Watson Blue Gene/L, XT3 and ANI’s Blue Gene/P.

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

<table>
<thead>
<tr>
<th>Cores</th>
<th>Default</th>
<th>Topology</th>
<th>Time per step</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>0.239</td>
<td>0.235</td>
<td>1.84</td>
</tr>
<tr>
<td>1024</td>
<td>0.189</td>
<td>0.182</td>
<td>1.91</td>
</tr>
<tr>
<td>2048</td>
<td>0.159</td>
<td>0.151</td>
<td>1.78</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Cores</th>
<th>Default</th>
<th>Topology</th>
<th>Time per step</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>0.217</td>
<td>0.215</td>
<td>2.20</td>
</tr>
<tr>
<td>1024</td>
<td>0.181</td>
<td>0.174</td>
<td>2.31</td>
</tr>
<tr>
<td>2048</td>
<td>0.154</td>
<td>0.147</td>
<td>2.39</td>
</tr>
</tbody>
</table>

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 Bhatele, Eric Bohm, and Laxmikant V. Kale

Details: http://charm.cs.uiuc.edu/~bhatele/charm/charm.html
Details: http://charm.cs.uiuc.edu/~bhatele/phd