Process Mapping

Mar 13, 2021

Topology-aware task mapping for reducing communication contention on large parallel machines

Tarun Agarwal, Amit Sharma, Laxmikant V. Kale

IPDPS 2006

Effect of Mapping

- 16 x 16 x 16 3D torus
  - Diameter: 24
- Average load on links high when #hops large

Message Size	Random Mapping	Optimal Mapping
1KB	56.93ms	46.91ms
10KB	243.64ms	124.56ms
100KB	2247.75ms	914.72ms
500KB	11.62s	4.44s
1MB	23.50s	8.80s

Table 1: Time for 200 iterations of a Jacobi-like program with optimal mapping and random mapping

Network Graphs

- Topology graph $G_p = (V_p, E_p)$
- Task graph $G_t = (V_t, E_t)$
  - c_{ab} → amount of communication (bytes) on e_{ab} between v_a and v_b

- Map: $v_t \in V_t$ placed on $v_p \in Vp$ $P : V_t \longrightarrow V_p$

Hop-bytes

Total size of inter-processor communication in bytes weighted by distance between the respective end-processors.

The overall hop-bytes is the sum of hop-bytes due to individual nodes in the task graph.

$$HB(G_t, G_p, P) = \frac{1}{2} \sum_{v_a \in V_t} HB(v_a)$$

$$where\ HB(v_a) = \sum_{e_{ab} \in E_t} HB(e_{ab})$$

$$where\ HB(e_{ab}) = c_{ab} \times d_p(P(v_a), P(v_b))$$

Hops per byte

Average number of network links a byte has to travel under a task mapping

$$Hops\ per\ Byte = \frac{HB(G_t, G_p, P)}{\sum_{e_{ab} \in E_t} c_{ab}}$$

$$Hops\ per\ Byte = \frac{\sum_{e_{ab} \in E_t} c_{ab} \times d_p(P(v_a), P(v_b))}{\sum_{e_{ab} \in E_t} c_{ab}}$$

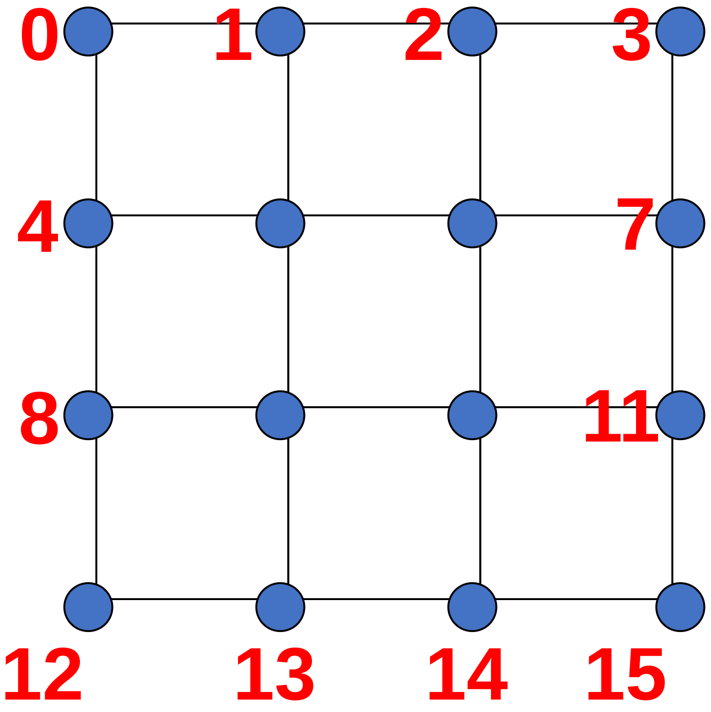
Process Mapping

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15

8 x 2 2D virtual process topology

Hop-bytes: bytes weighed by distance

Hops per byte: average links traversed by a byte

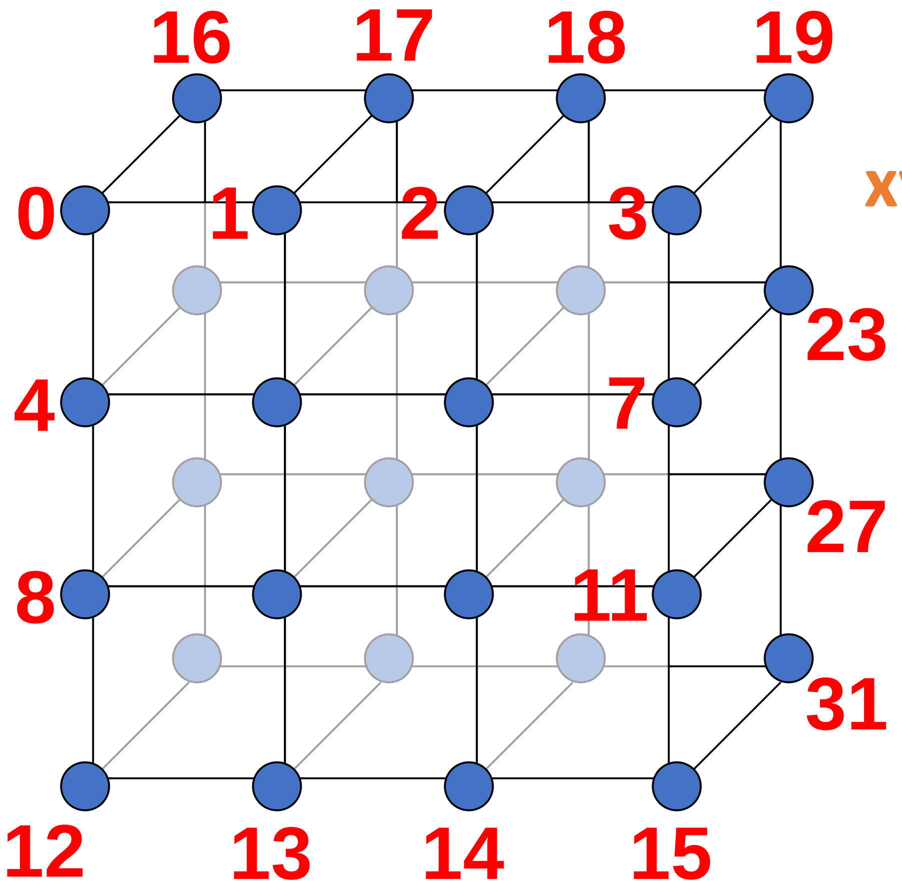


4 x 4 2D mesh

Process Mapping

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31

8 x 4 2D virtual process topology



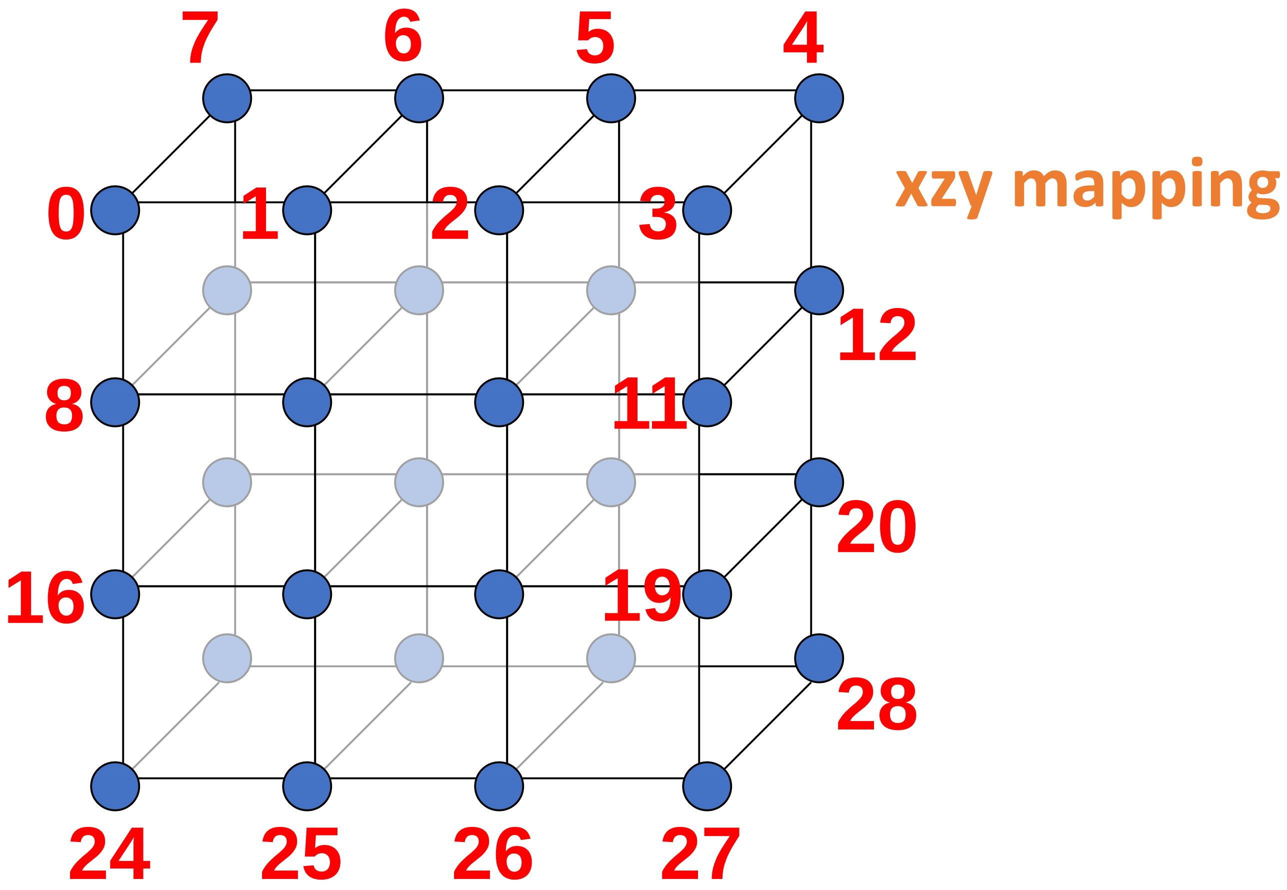
xyz mapping

4 x 4 x 2 3D torus

Topology-aware Process Mapping

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31

8 x 4 2D virtual process topology



4 x 4 x 2 3D torus

This Paper

A process mapping strategy that minimizes the impact of topology by heuristically minimizing the total number of hop-bytes communicated

Two Phases

- Partitioning
    - Partitioning compute objects into p groups (assume p processors)
    - METIS*
- Mapping
    - Map the p groups to p processors such that the heavily communicating groups are placed on nearby processors
    - Where should the next process be placed?

*"A fast and high quality multilevel scheme for partitioning irregular graphs", George Karypis and Vipin Kumar, International Conference on Parallel Processing (ICPP), pp. 113-122, 1995

Estimation Function

- $f_{est}(t, p, M) =$ Cost of estimating the placement of a task t onto processor p under current task mapping M
- Estimate how critical it is to place a task in the current cycle, select the task with maximum criticality
- T_k is the set of tasks yet to be placed
- P_k is the set of processors that are available

$T_k \cup \bar{T}_k = \emptyset$
$P_k \cup \bar{P}_k = \emptyset$

Estimation Function

- $f_{est}(t, p, M)$ = Cost of estimating the placement of a task t onto processor p under current task mapping M
- First order approximation w.r.t placed tasks
  - Hop-bytes

$$f_{est}(t_i, p, M) = \sum_{t_j \in \bar{T}_k} c_{ij} d_p(p, M(t_j))$$

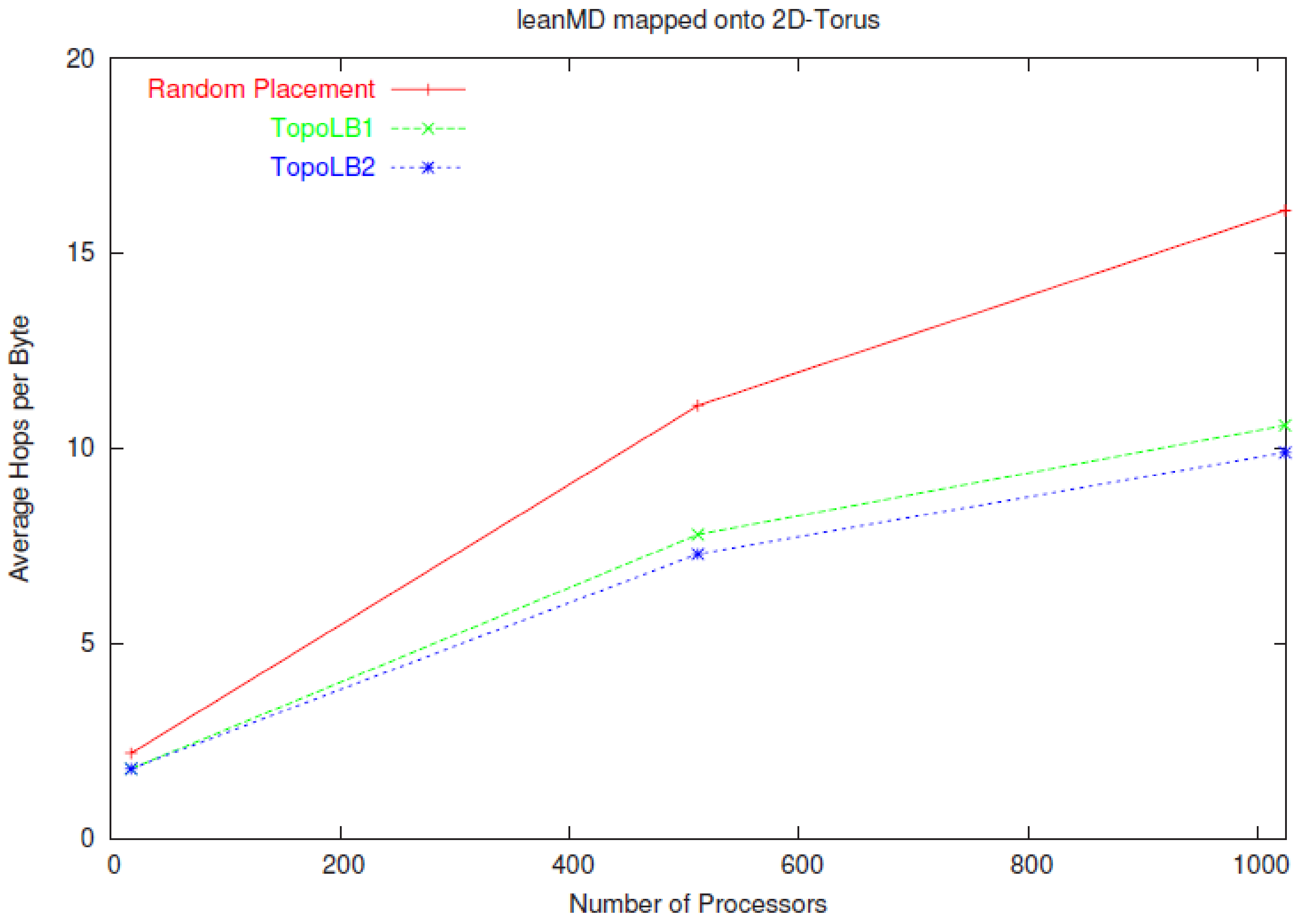
- Second order approximation w.r.t all tasks

$$d_p(p, M(t_j)) \approx \frac{\sum_{p_j \in V_p} d_p(p, p_j)}{|V_p|}$$

Mapping Heuristic

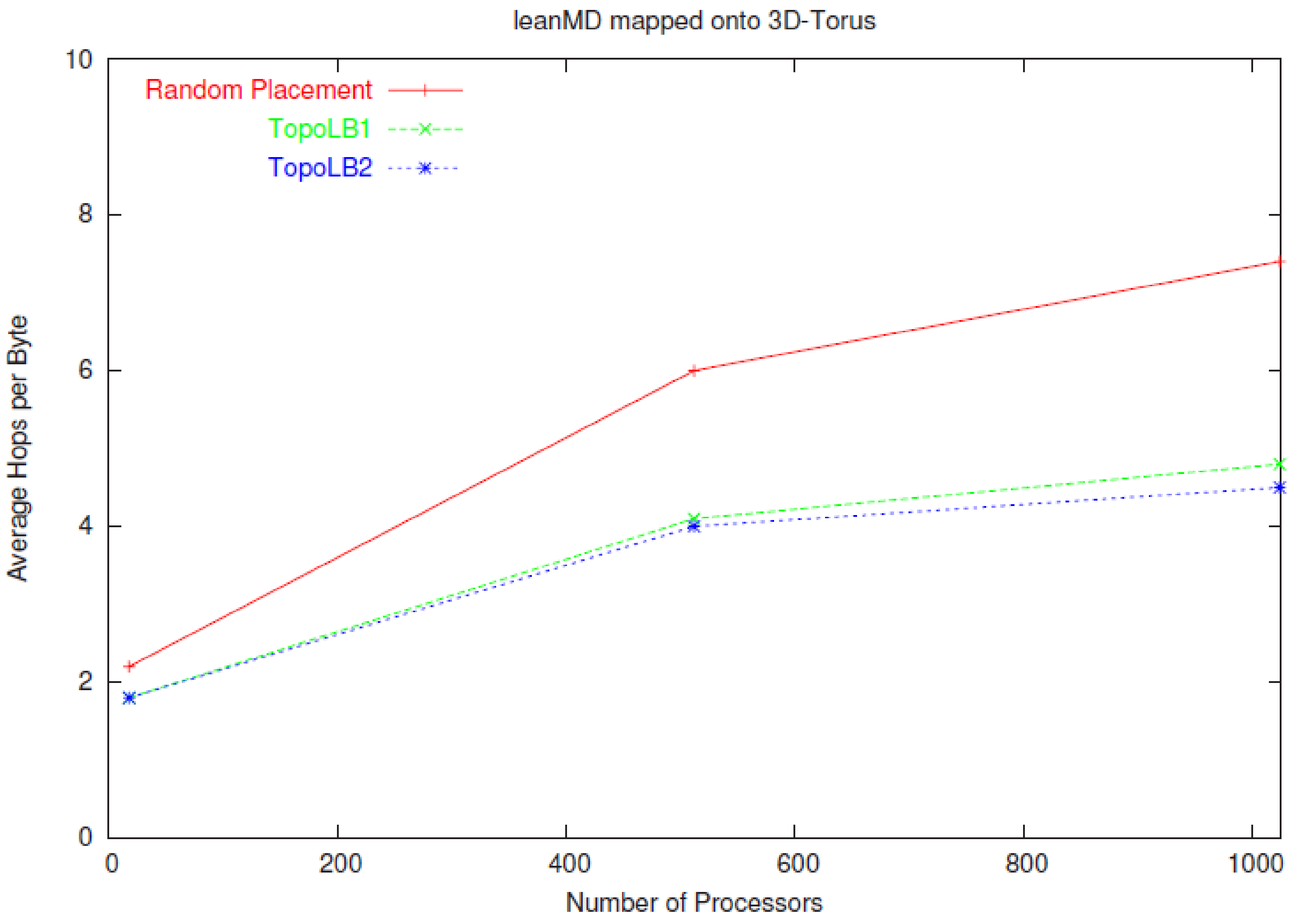
- For each task, find the best processor, the one where it costs least to place it.
- For each task, the estimation function gives the cost of placing a task on its best processor and the expected cost when placed on an arbitrary processor.
- Determine how critical it is to place the task in the current cycle and select the most critical task for placement in the current cycle.
- The estimation function f (using hop-bytes) is maintained in a matrix $\mathrm{p \ x \ p}$.
- Select the task that maximizes $f_{avg}[t] - fm_{in}[t]$.
- Find the processor that minimizes f.

Results – LeanMD on 2D Torus



34% reduction
in hops-per-byte

Results – LeanMD on 3D Torus



40% reduction in hops-per-byte