

Yi-shui LI, Xin-hai CHEN, Jie LIU, Bo YANG, Chun-ye GONG, Xin-biao GAN, Sheng-guo LI, Han XU, 2020. OHTMA: an optimized heuristic topology-aware mapping algorithm on the Tianhe-3 exascale supercomputer prototype. *Frontiers of Information Technology & Electronic Engineering*, 21(6):939-949. <https://doi.org/10.1631/FITEE.1900075>

OHTMA: an optimized heuristic topology-aware mapping algorithm on the Tianhe-3 exascale supercomputer prototype

Key words: High-performance computing; Topology mapping; Heuristic algorithm

Corresponding author: Jie LIU

E-mail: liujie@nudt.edu.cn

 ORCID: Jie LIU, <https://orcid.org/0000-0003-3745-7541>

Motivation

With the scale of high-performance computing (HPC) growing rapidly, the constraints of communication performance become evident when large-scale applications run on HPC systems because of the increased traffic, but the restricted development of network bandwidth cannot catch up with the current requirement of large-scale systems.

Congestion is a dominant factor that can significantly affect communication performance, and a refined mapping of application tasks can effectively reduce this congestion.

Main idea

The main idea of this algorithm is to generate a greedy primary mapping strategy, use numerous pair-exchange operations to minimize the evaluation metric within a finite number of iterations, and backtrack to the best result.

We optimize this new algorithm to minimize its runtime, which can improve its practicability.

Method

1. Topology of the Tianhe-3 exascale supercomputer prototype

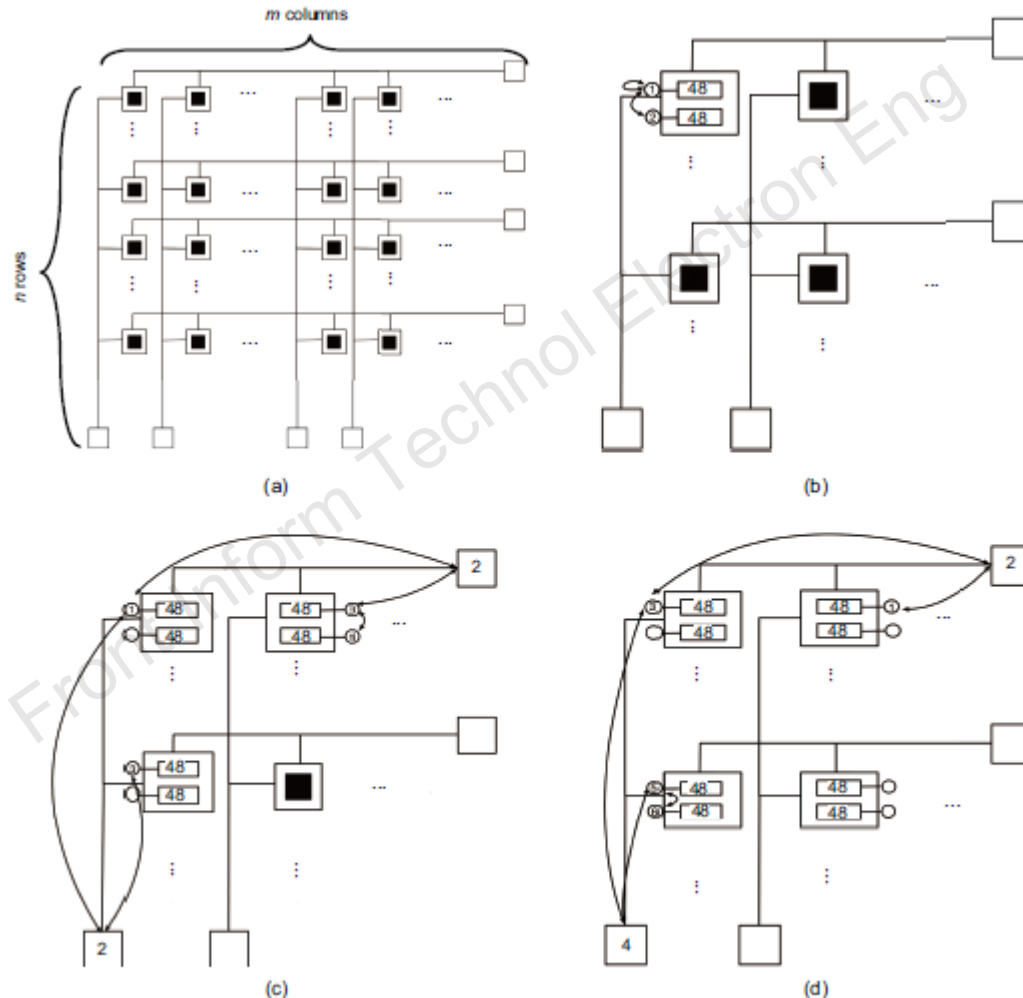


Fig. 1 Topology of the Tianhe-3 exascale supercomputer prototype (a) and three hop situations: (b) two compute nodes within a chip; (c) two compute nodes within a same row or column; (d) two compute nodes in different rows and columns

2. OHTMA pseudocode

Algorithm 1 Optimized heuristic topology-aware mapping algorithm (OHTMA) on Tianhe-3

Require: process P , idle processor N , communication pattern matrix $\mathbf{A}$, and hop matrix $\mathbf{B}$

Ensure: final mapping result f

```
// Begin the initial part
// At the beginning, all the processes and processors are unselected
1:  $P_{\text{selected}} \leftarrow \emptyset, N_{\text{selected}} \leftarrow \emptyset$ 
2:  $P_{\text{unselected}} \leftarrow P, N_{\text{unselected}} \leftarrow N$ 
3:  $k \leftarrow 0$ 
4: while  $k < |P|$  do
5:   #pragma omp parallel for
6:   for each process  $p$  in  $P_{\text{unselected}}$  do
7:      $\text{comm}_p \leftarrow \sum_{i \in P_{\text{selected}}} A_{pi} + \frac{\sum_{j \in P_{\text{unselected}}} A_{pj}}{1 + |P_{\text{selected}}|}$ 
8:   end for
9:   #pragma omp parallel for
10:  for each processor  $n$  in  $N_{\text{unselected}}$  do
11:     $\text{hops}_n \leftarrow \sum_{i \in N_{\text{selected}}} B_{ni} + \frac{\sum_{j \in N_{\text{unselected}}} B_{nj}}{1 + |N_{\text{selected}}|}$ 
12:  end for
13:  // Select the processes with the maximum  $\text{comm}_p$  and the processors with the minimum number of hops
14:   $p_{\text{max}} \leftarrow \max \{ \text{comm}_p \}, p \in P_{\text{unselected}}$ 
15:   $n_{\text{min}} \leftarrow \min \{ \text{hops}_n \}, n \in N_{\text{unselected}}$ 
16:  // Update mapping  $f$  and the subsets
17:   $f(p_{\text{max}}) \leftarrow n_{\text{min}}$ 
18:   $P_{\text{selected}} \leftarrow P_{\text{selected}} + \{p_{\text{max}}\}, N_{\text{selected}} \leftarrow N_{\text{selected}} + \{n_{\text{min}}\}$ 
19:   $P_{\text{unselected}} \leftarrow P_{\text{unselected}} - \{p_{\text{max}}\}, N_{\text{unselected}} \leftarrow N_{\text{unselected}} - \{n_{\text{min}}\}$ 
20:   $k \leftarrow k + 1$ 
21: end while
22: // Begin mapping optimization
```

2. OHTMA pseudocode

```
20: Status[|P|]  $\leftarrow 0, k \leftarrow 0$ 
21: // Loop is a user-defined number of iterations
22: while  $k < \text{loop}$  do
23:   #pragma omp parallel for
24:   for each  $i, j \in P$ , where  $\text{status}[i] \neq 1 \ \&\& \ \text{status}[j] \neq 1$  do
25:     // Calculate the hop-byte changes after exchanging these two processes
26:      $\text{Cost}_{ij} = \sum_{l \in P} A_{il} B_{f(i)f(l)} + A_{jl} B_{f(j)f(l)} - \sum_{r \in P} A_{ir} B_{f(j)f(r)} + A_{jr} B_{f(i)f(r)}$ 
27:   end for
28:   // Find the maximum element in the Cost matrix
29:    $\text{Cost}_{mn} \leftarrow \max \{ \mathbf{Cost} \}$ 
30:   // Pair[ ] is used to store the exchanged pair and Result[ ] is used to store the change of hop-byte
31:    $\text{Pair} \leftarrow \text{Pair} + \{(m, n)\}, \text{Result} \leftarrow \text{Result} + \{\text{Cost}_{mn}\}$ 
32:    $\text{Status}[|P|] \leftarrow 1, k \leftarrow 1$ 
33:   Exchange  $f(m)$  and  $f(n)$ 
34: end while
35: // Calculate the sum of the first  $n$  terms ( $n$  from 0 to |Result|) in Result[ ]
36:  $\text{Sum}[ ] \leftarrow \text{sum}(\text{Result}[ ])$ 
37: // Find index  $t$  of the maximum value in Sum[ ]
38:  $t \leftarrow \max(\text{Sum}[ ])$ 
39: Backtrack to the first  $t$  exchanges and obtain the final mapping  $f$ 
40: Return  $f$ 
```

Major results

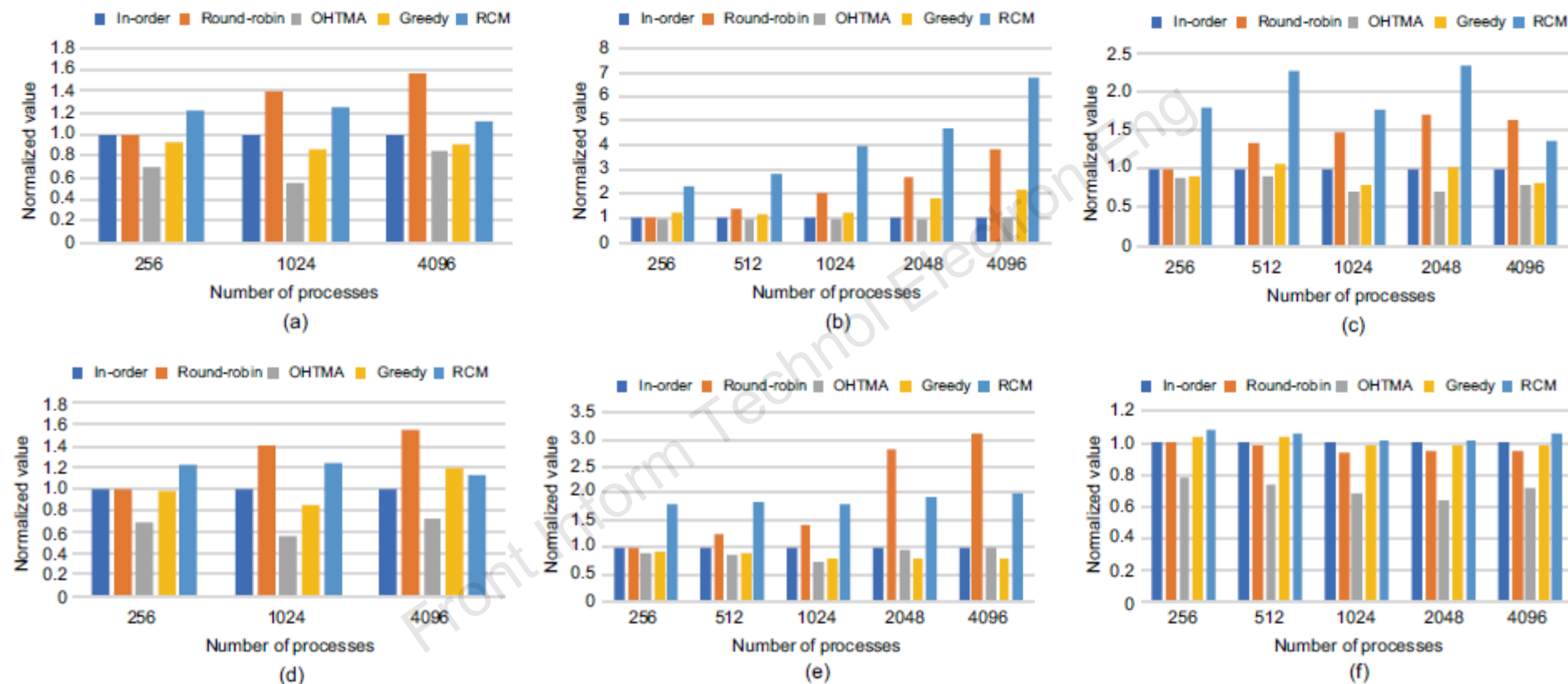


Fig. 2 Normalized cost metric for various mapping algorithms on Tianhe-3: (a) BT; (b) CG; (c) LU; (d) SP; (e) Sweep3D; (f) Snap

BT: block tri-diagonal solver; CG: conjugate gradient, irregular memory access, and communication; LU: lower-upper Gauss-Seidel solver; SP: scalar penta-diagonal solver

Major results

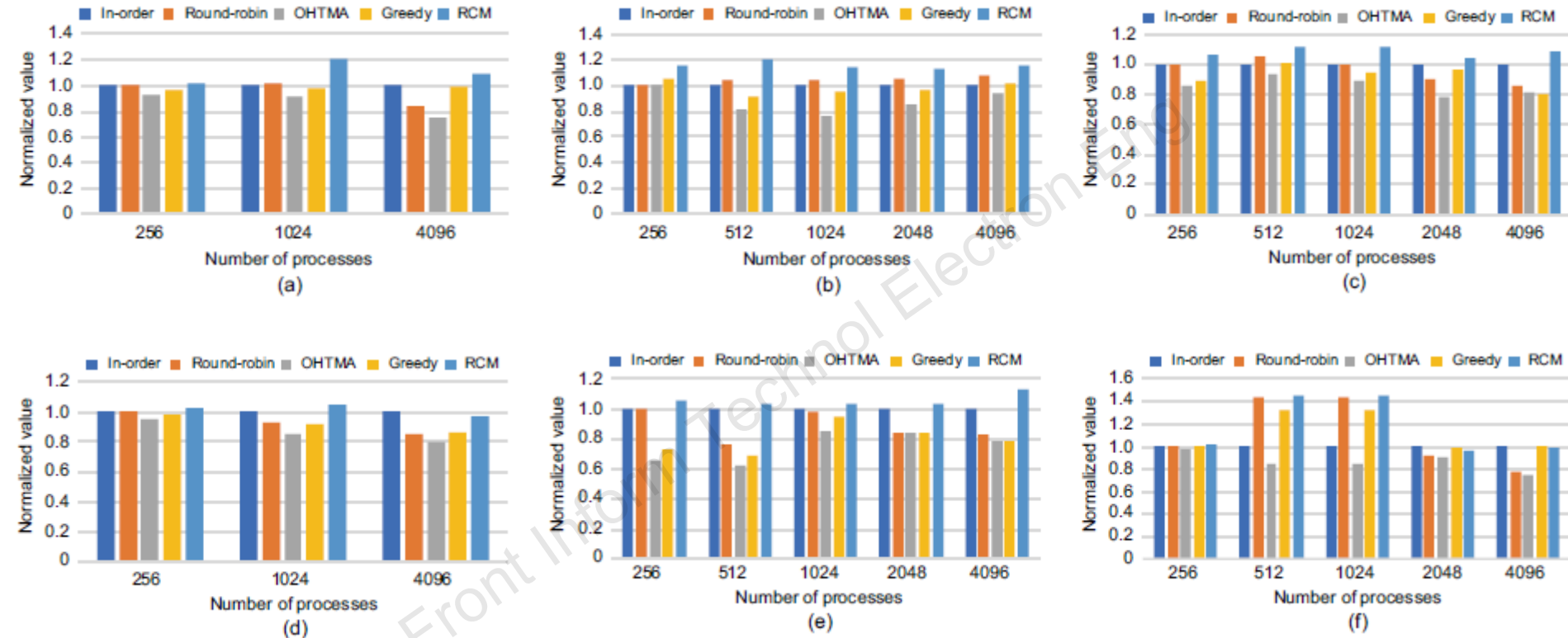


Fig. 3 Normalized communication time for various mapping algorithms on Tianhe-3: (a) BT; (b) CG; (c) LU; (d) SP; (e) Sweep3D; (f) Snap

BT: block tri-diagonal solver; CG: conjugate gradient, irregular memory access, and communication; LU: lower-upper Gauss-Seidel solver; SP: scalar penta-diagonal solver

Conclusions

We have proposed a new optimized heuristic topology-aware mapping algorithm (OHTMA). OHTMA provides a greedy mapping strategy and imports the pair-exchange method in the mapping optimization operation.

OHTMA has been applied on the Tianhe-3 exascale supercomputer prototype. Two default mapping algorithms and two typical mapping algorithms have been compared, and OHTMA has obtained satisfactory experimental improvements. OHTMA can effectively reduce both the hop-byte value and the communication time.