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

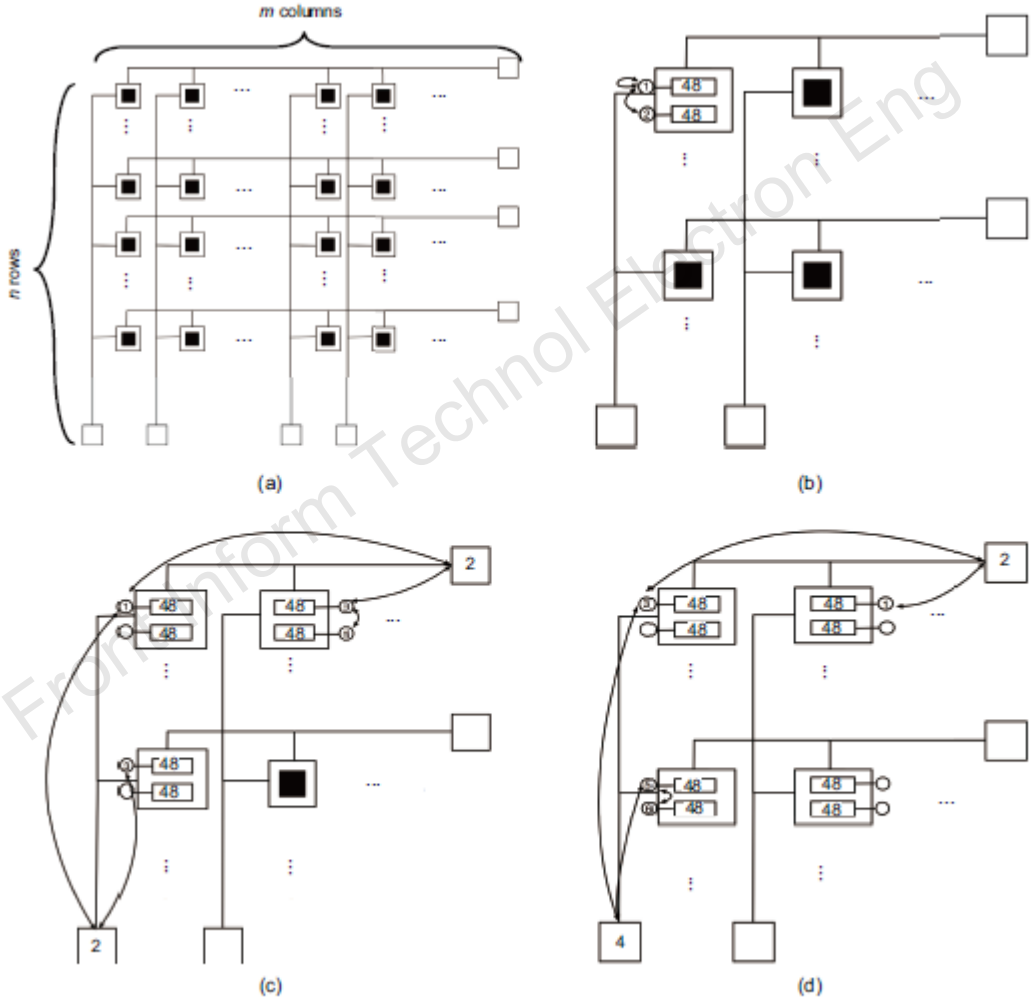

Fig. 1 Topology of the Tianhe-3 exascale supercomputer prototype (a) and three hop situations: (b) two compute nodes within a chip; (c) two compute nodes within a same row or column; (d) two compute nodes in different rows and columns

2. OHTMA pseudocode

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

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

Ensure: final mapping result f

// Begin the initial part

// At the beginning, all the processes and processors are unselected

$P_{\text{selected}} \leftarrow \varnothing$, $N_{\text{selected}} \leftarrow \varnothing$

$P_{\text{unselected}} \leftarrow P$, $N_{\text{unselected}} \leftarrow N$

$k \leftarrow 0$

while $k < |P|$ **do**

  #pragma omp parallel for

  for each process p in $P_{\text{unselected}}$ **do**

    $\text{comm}_p \leftarrow \sum_{i \in P_{\text{selected}}} A_{pi} + \dfrac{\sum_{j \in P_{\text{unselected}}} A_{pj}}{1 + |P_{\text{selected}}|}$

  end for

  #pragma omp parallel for

  for each processor n in $N_{\text{unselected}}$ **do**

    $\text{hops}_n \leftarrow \sum_{i \in N_{\text{selected}}} B_{ni} + \dfrac{\sum_{j \in N_{\text{unselected}}} B_{nj}}{1 + |N_{\text{selected}}|}$

  end for

  // Select the processes with the maximum comm_p and the processors with the minimum number of hops

  $p_{\max} \leftarrow \max\{\text{comm}_p\}$, $p \in P_{\text{unselected}}$

  $n_{\min} \leftarrow \min\{\text{hops}_n\}$, $n \in N_{\text{unselected}}$

  // Update mapping f and the subsets

  $f(p_{\max}) \leftarrow n_{\min}$

  $P_{\text{selected}} \leftarrow P_{\text{selected}} + \{p_{\max}\}$, $N_{\text{selected}} \leftarrow N_{\text{selected}} + \{n_{\min}\}$

  $P_{\text{unselected}} \leftarrow P_{\text{unselected}} - \{p_{\max}\}$, $N_{\text{unselected}} \leftarrow N_{\text{unselected}} - \{n_{\min}\}$

  $k \leftarrow k + 1$

end while

// 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** $k <$ loop **do**
23: #pragma omp parallel for
24: **for** each $i, j \in P$, where status[i] $\neq 1$ && 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**
// Find the maximum element in the **Cost** matrix
28: $\text{Cost}_{mn} \leftarrow \max\{\textbf{Cost}\}$
// Pair[ ] is used to store the exchanged pair and Result[ ] is used to store the change of hop-byte
29: Pair $\leftarrow$ Pair $+ \{(m, n)\}$, Result $\leftarrow$ Result $+ \{\text{Cost}_{mn}\}$
30: Status[$|P|$] $\leftarrow 1, k \leftarrow 1$
31: Exchange $f(m)$ and $f(n)$
32: **end while**
// Calculate the sum of the first n terms (n from 0 to |Result|) in Result[ ]
33: Sum[ ] $\leftarrow$ sum(Result[ ])
// Find index t of the maximum value in Sum[ ]
34: $t \leftarrow \max$(Sum[ ])
35: Backtrack to the first t exchanges and obtain the final mapping f
36: Return f

Major results

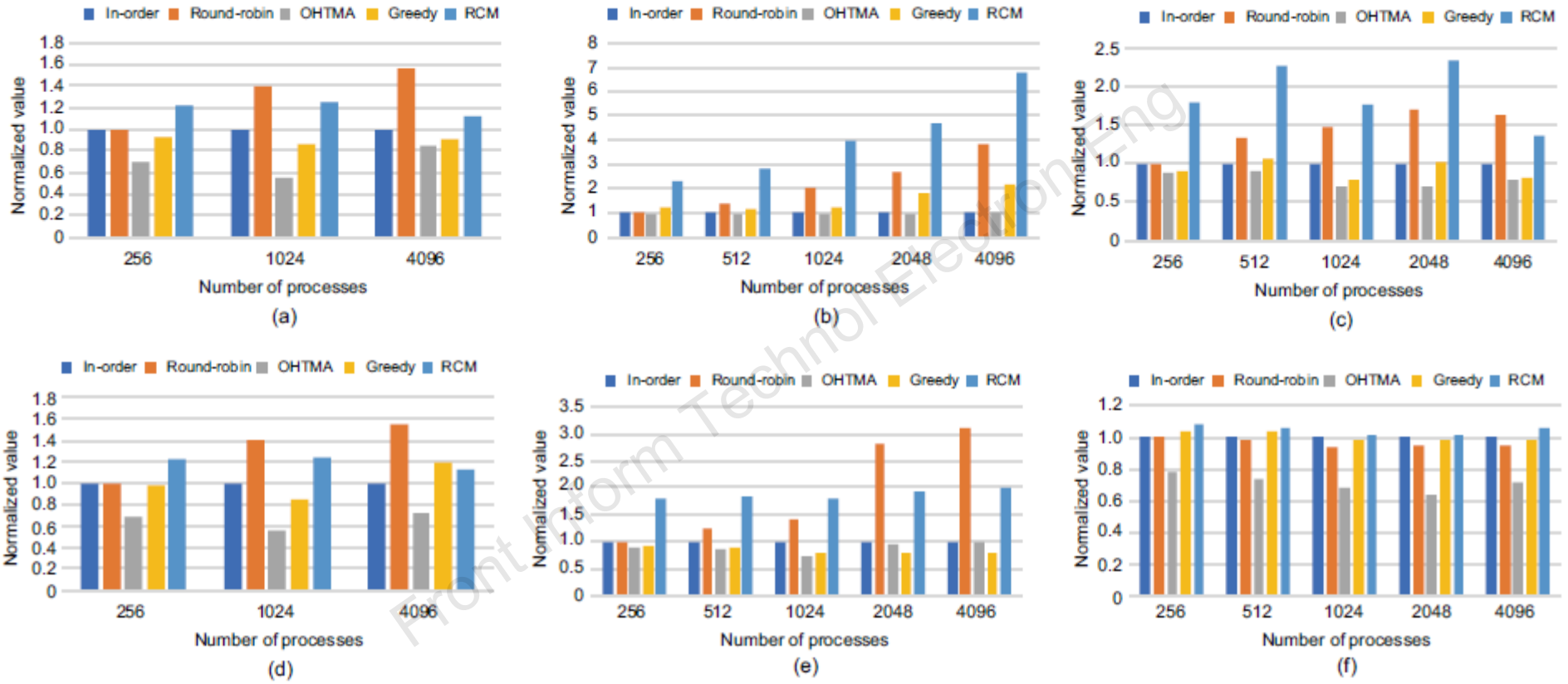

Fig. 2 Normalized cost metric for various mapping algorithms on Tianhe-3: (a) BT; (b) CG; (c) LU; (d) SP; (e) Sweep3D; (f) Snap

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

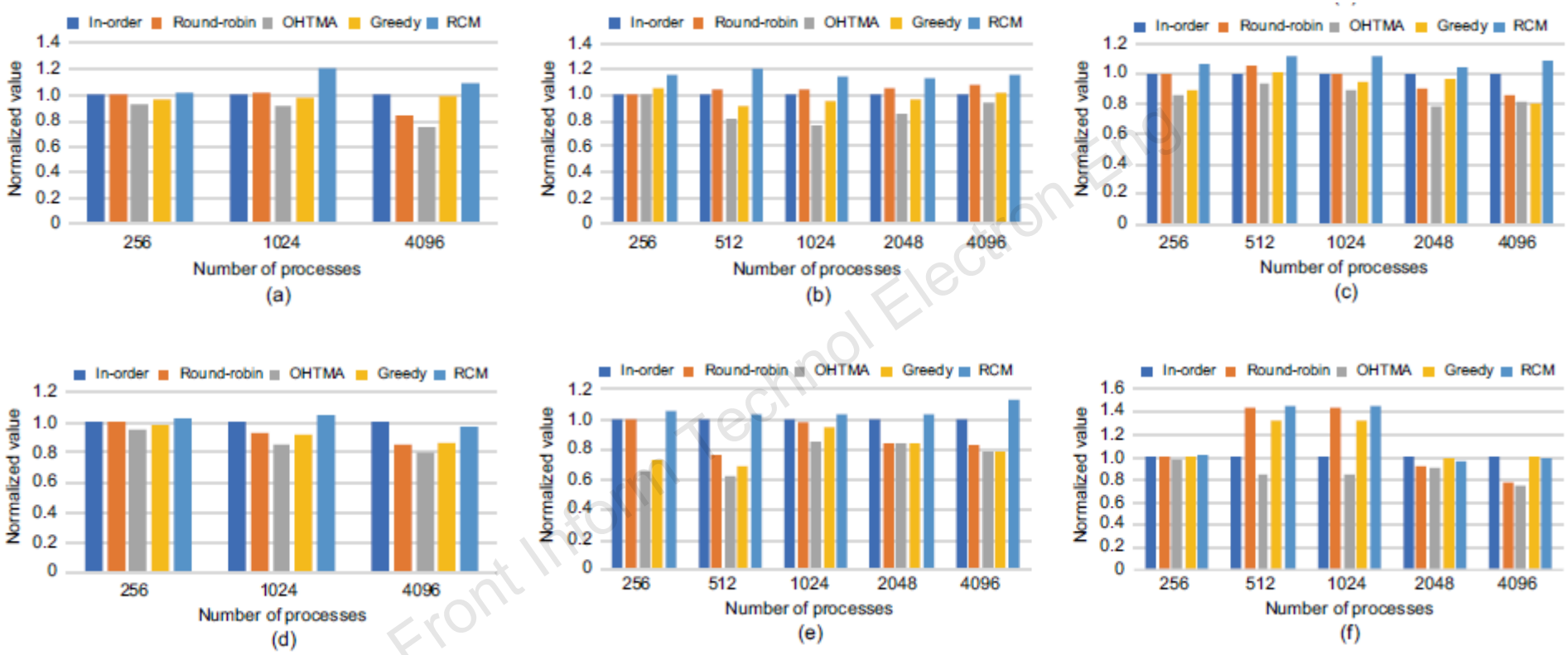

Fig. 3 Normalized communication time for various mapping algorithms on Tianhe-3: (a) BT; (b) CG; (c) LU; (d) SP; (e) Sweep3D; (f) Snap

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.