OPTIMIZATION OF COMMUNICATION AND MEMORY ACCESS IN BFS BENCHMARK

March 5, 2015

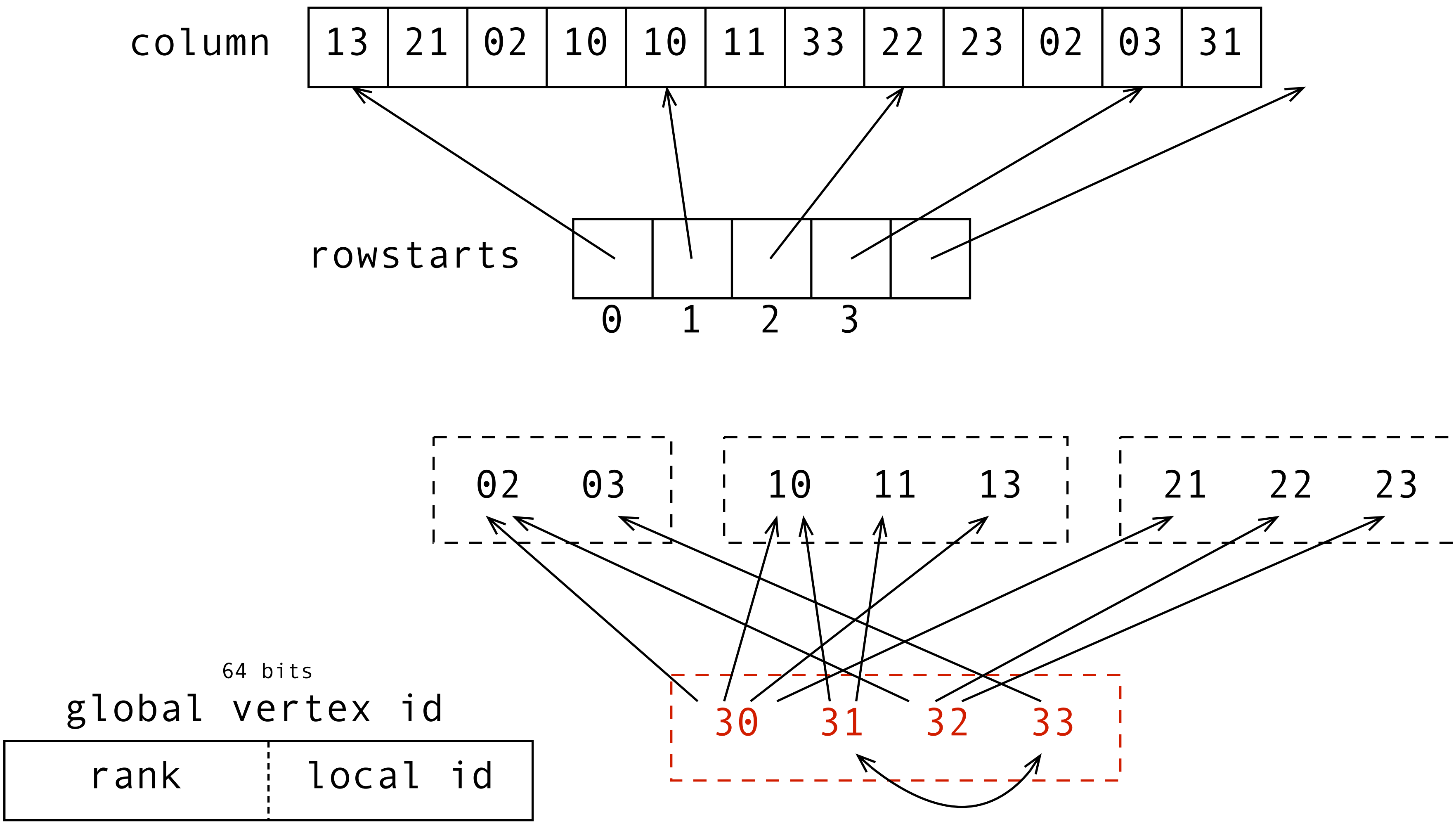
Alexander Daryin
Head of Advanced Research Lab
alexander.daryin@t-platforms.ru

Introduction

- **Breadth-First Search (BFS)** on distributed memory systems
- Part of Graph500 benchmark

- Performance is limited by
  - memory access latency
  - network latency
  - network bisection bandwidth

Compressed Sparse Row Graph (CSR)



Reference Algorithm

Data: CSR (row starts R, column C), current queue Q (array)

function `ProcessQueue(R, C, Q)`
- **for** $v \in \mathsf{Q}$ **do**
  - **for** $e \leftarrow \mathsf{R}[v]$ **to** $\mathsf{R}[v+1]$ **do**
    - `send` v, `local`($\mathsf{C}[e]$) `to owner`($\mathsf{C}[e]$)

Reference Algorithm

Data: CSR (row starts R, column C), current queue Q (array)

```
function ProcessQueue(R, C, Q)
    for v ∈ Q do
        for e ← R[v] to R[v + 1] do
            send v, local(C[e]) to owner(C[e]))
```

Random memory access pattern

cache miss

Reference Algorithm

Data: CSR (row starts R, column C), current queue Q (array)

```
function ProcessQueue(R, C, Q)
    for v ∈ Q do
        for e ← R[v] to R[v + 1] do
            send v, local(C[e]) to owner(C[e])
```

Small message size
low bandwidth

Irregular communication pattern
connection overhead

Message coalescing

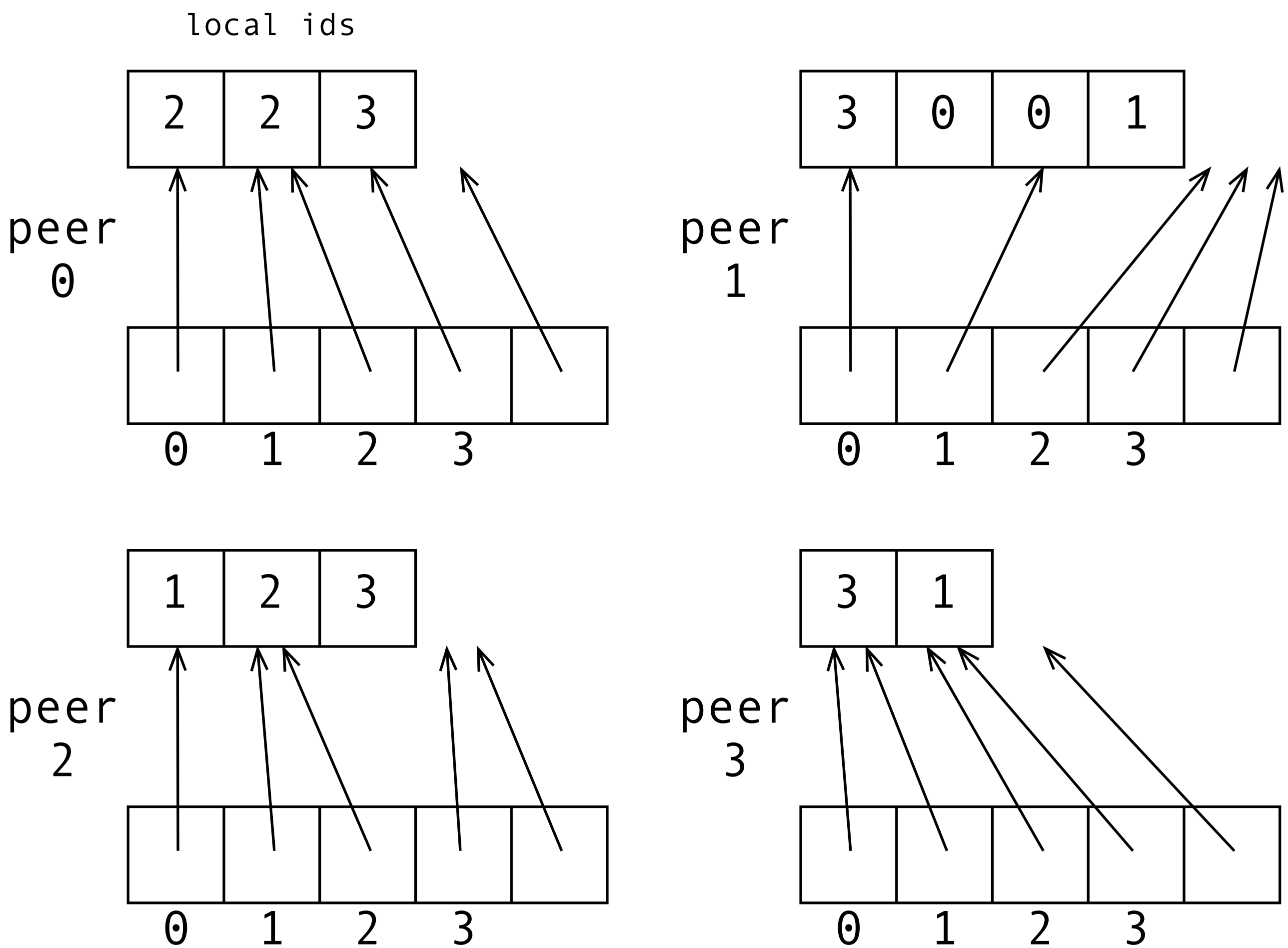
Separate send buffer for each peer process
large memory overhead
multiple threads - ?

Goals

- Optimize memory access
- Regularize communication pattern
- Reduce memory consumption
- Allow for multithreaded design

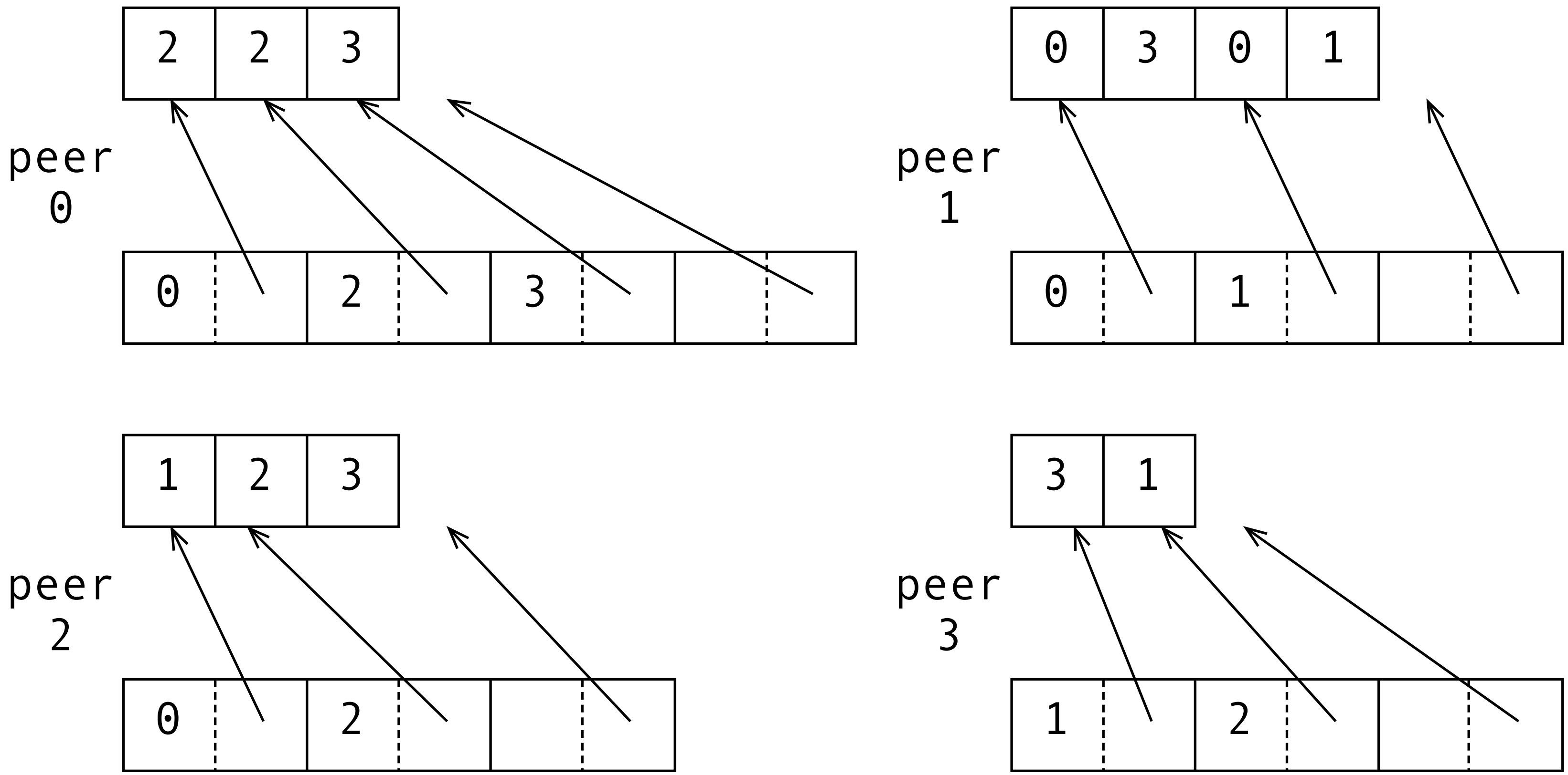
- **Possible solution:**
  - partition graph data for each peer node
  - process partitions independently (and possibly concurrently)

Naïve Partitioning

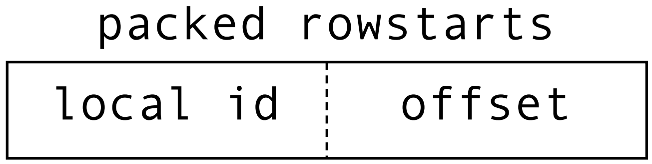


- **Infeasible:** |rowstarts| = |V|

Packed Partitioned CSR



- |rowstarts| ≤ |local edges| + NP

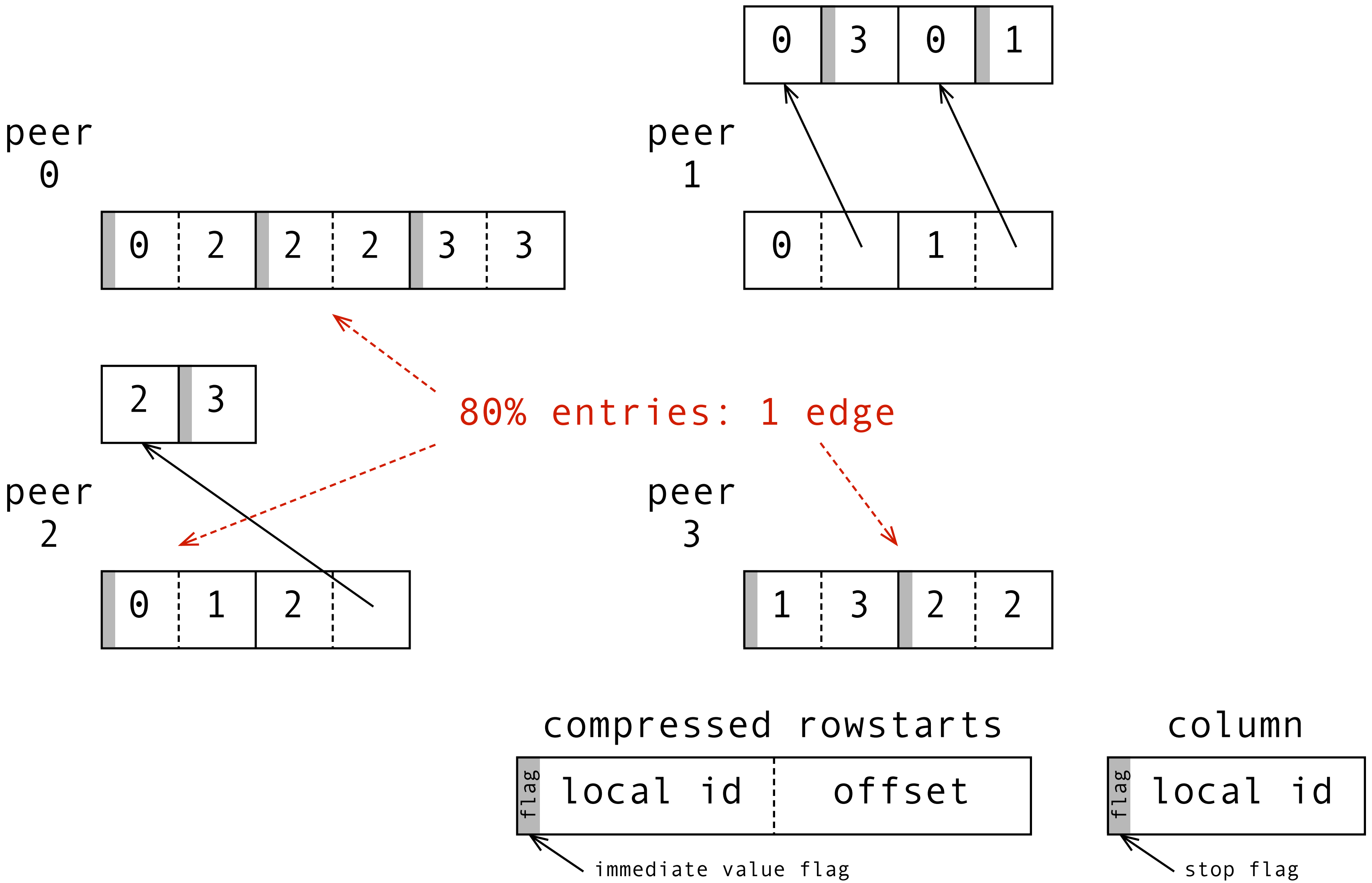


Algorithm for Packed CSR

Data: Packed CSR (vertex indices V, row offsets D, column C), current queue Q (bit mask), number of processes NP

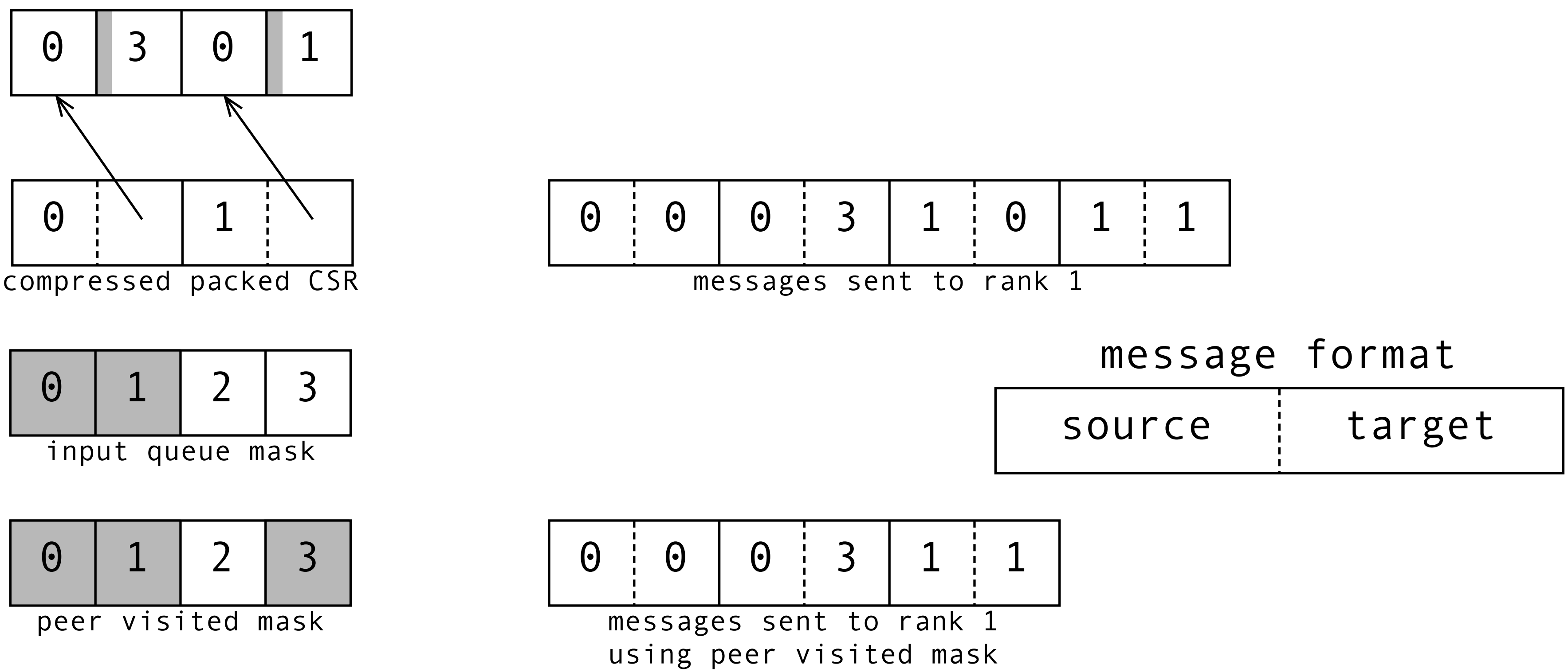
```
function ProcessQueue()
    for peer ← 0 to NP do
        for i ← 0, |V[peer]| do
            if V[peer][i] ∈ Q then
                for e ← D[peer][i] to D[peer][i + 1] do
                    send V[peer][i], C[e] to peer
```

Compressed Packed CSR



Peer Visited Mask

- Do not repeat target vertices (in current round)
- **2x speedup**
- Impossible with original non-partitioned CSR



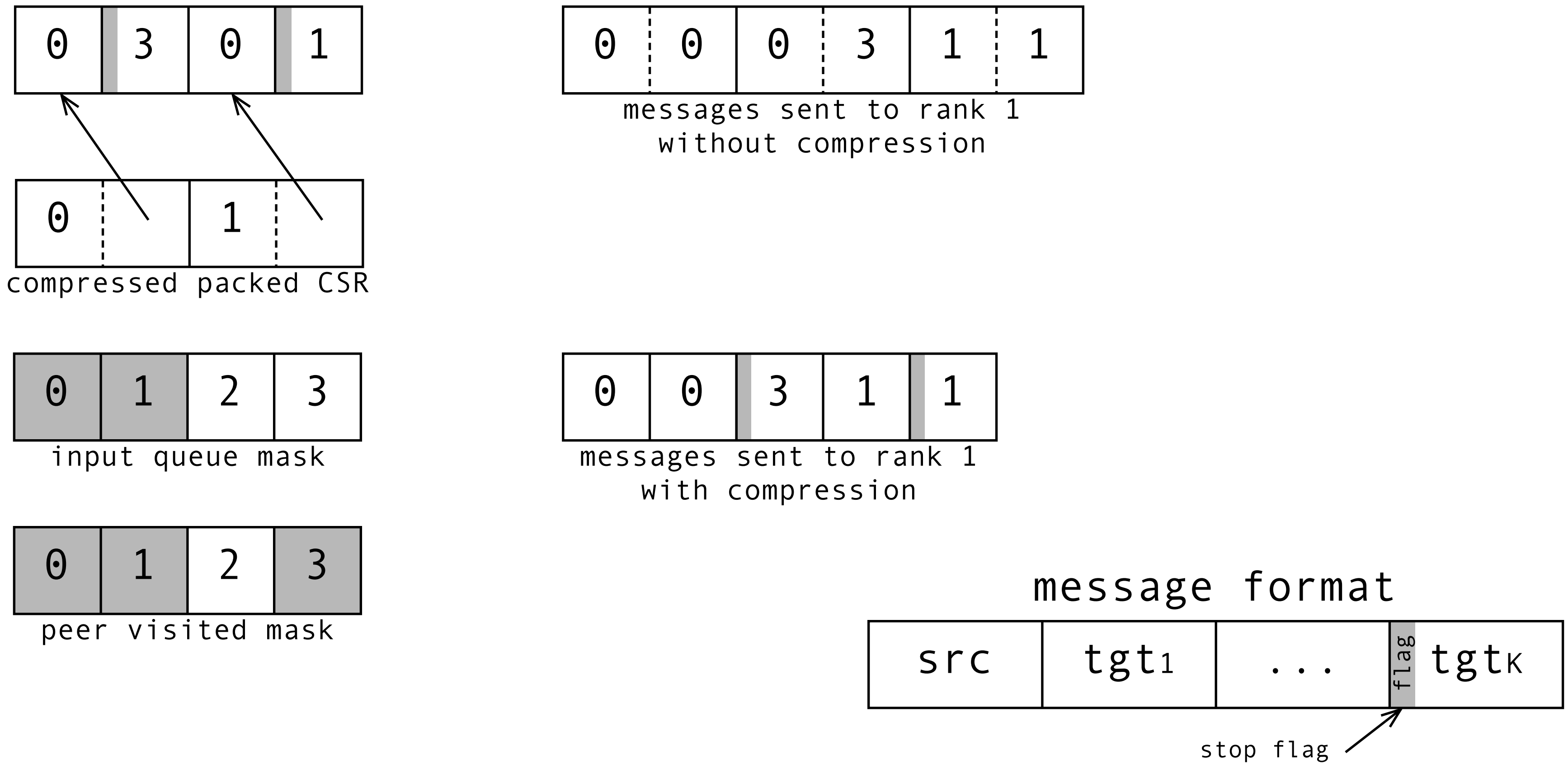
Algorithm with Peer Visited Mask

Data: Packed CSR (vertex indices V, row offsets D, column C), current queue Q (bit mask), number of processes NP

function `ProcessQueue()`
    for peer $\leftarrow 0$ **to** NP **do**
        `clear(P)`   `/* clear peer visited mask */`
        for $i \leftarrow 0, |\mathsf{V}[\mathsf{peer}]|$ **do**
            if $\mathsf{V}[\mathsf{peer}][i] \in \mathsf{Q}$ **then**
                for $e \leftarrow \mathsf{D}[\mathsf{peer}][i]$ **to** $\mathsf{D}[\mathsf{peer}][i+1]$ **do**
                    if $\mathsf{C}[e] \notin \mathsf{P}$ **then**
                        `set` $\mathsf{P}[\mathsf{C}[e]]$
                        `send` $\mathsf{V}[\mathsf{peer}][i]$, $\mathsf{C}[e]$ `to peer`

Message Compression

- Do not repeat source vertices
- **Up to 40% traffic reduction** (in forward phase)



Benefits of Packed CSR

- **Sequential memory access**
- **Regular communication pattern**
  - Fixed number of large send/receive buffers
  - Lower overhead incurred by network adapters
  - Use dynamically connected transport (DC/XRC) on modern InfiniBand hardware
- **Reduced network traffic**
- **Individual CSRs may be processed concurrently**

Additional Optimizations

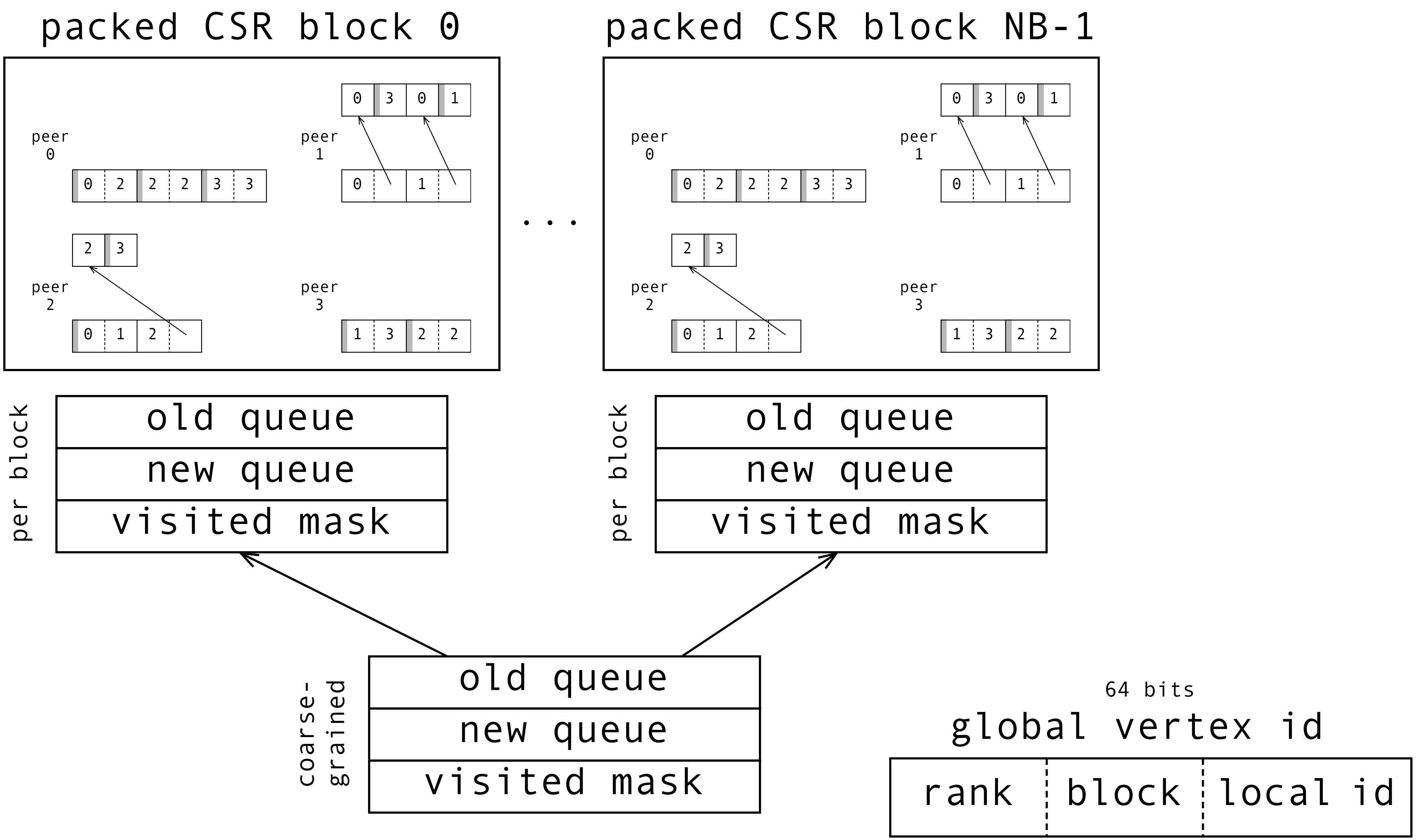
- **Direction Optimization** (backward stepping)
- Blocks of packed CSRs:
  - **Coarse-grained queue mask** (forward phase)
  - **Coarse-grained visited mask** (backward phase)
- **Load balancing** (heavy vertices)

Backward Stepping

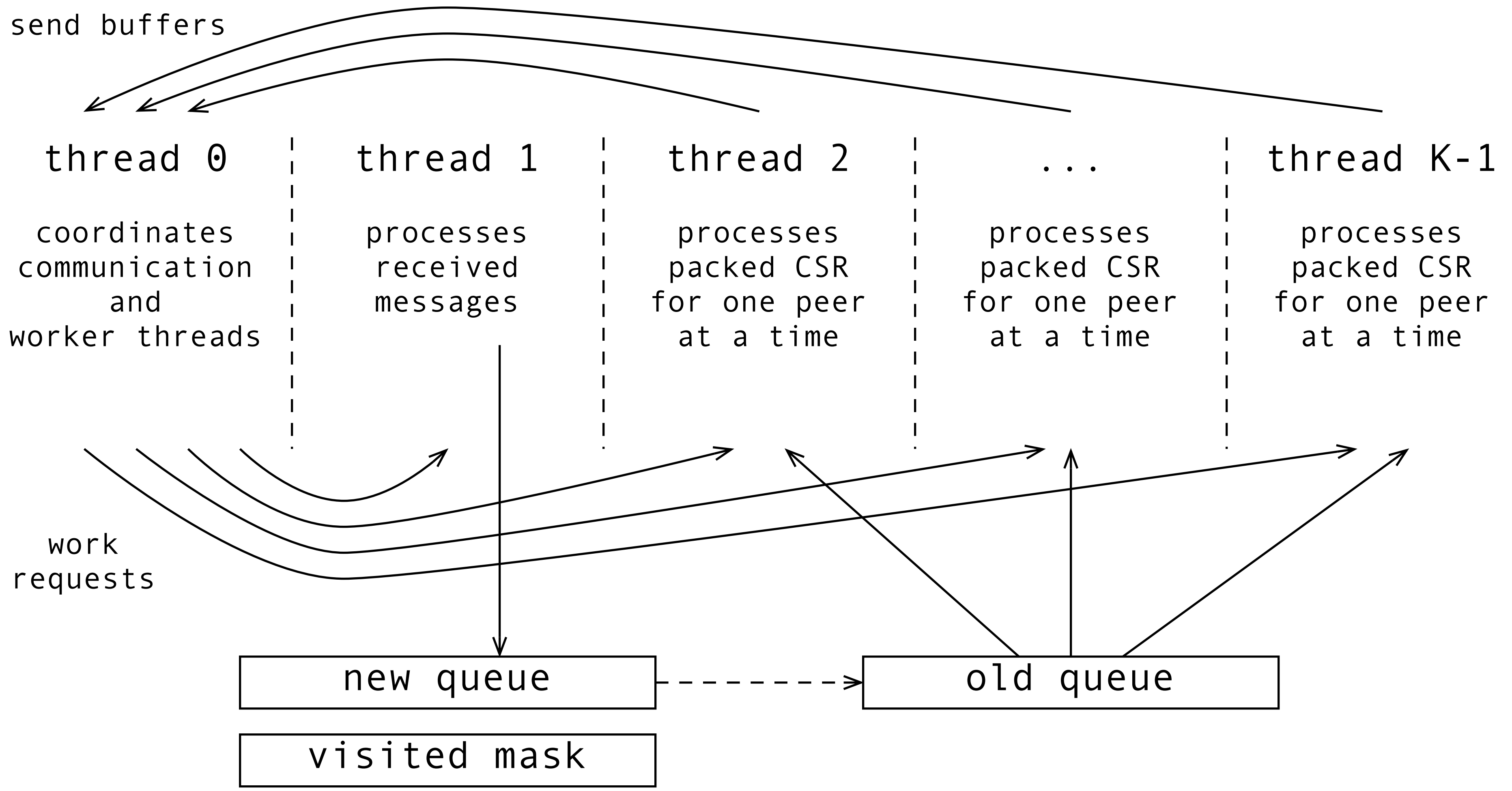
Data: Packed CSR (vertex indices V, row offsets D, column C), bit mask of visited vertices M, number of processes NP

```
function ProcessUnvisited()
    // probe first edges
    for peer ← 0 to NP do
        for i ← 0, |V[peer]| do
            if V[peer][i] ∉ M then
                send V[peer][i], C[D[peer][i]] to peer
        flush send buffer
    wait for all acks
    // probe other edges
    for peer ← 0 to NP do
        for i ← 0, |V[peer]| do
            if V[peer][i] ∉ M then
                for e ← D[peer][i] + 1 to D[peer][i + 1] do
                    send V[peer][i], C[e] to peer
        flush send buffer
```

Packed CSR Blocks

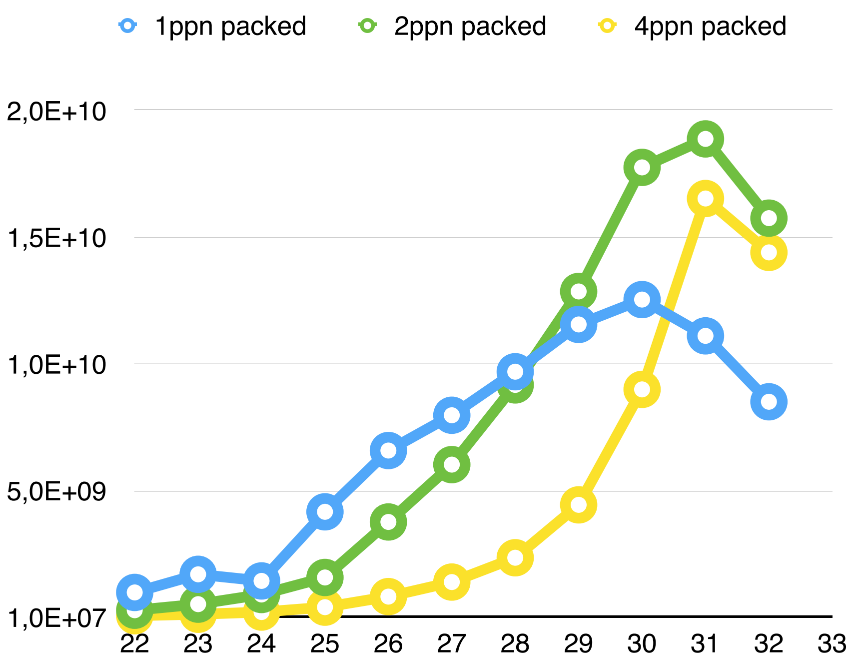


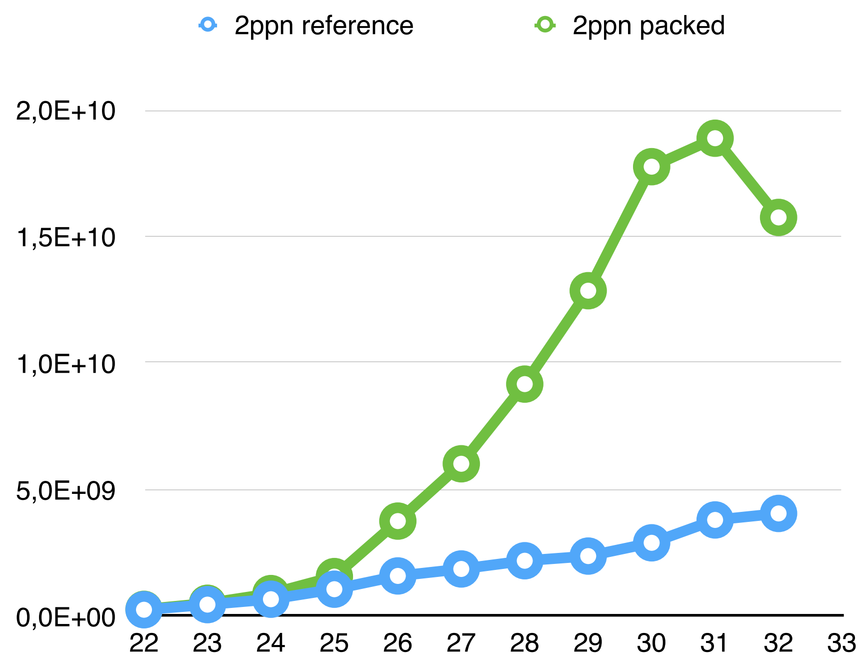
Multithreading (Work in Progress)



Performance: Lomonosov

- 512 nodes in partition regular4, InfiniBand QDR





Performance: Lomonosov-2

- 64 nodes in R&D partition, InfiniBand FDR