Engineering a Distributed-Memory Triangle Counting Algorithm

IPDPS 2023
Peter Sanders and Tim Niklas Uhl | May 18, 2023
Problem Definition

- **Given:** undirected graph $G = (V, E)$
- **Output:** all triangles $\{u, v, w\} \subseteq V$ with $\{u, v\}, \{v, w\}, \{u, w\} \in E$.

Applications

- graph analysis $\rightarrow$ local clustering coefficient
- spam detection, link recommendation, ...

**Goal:** distributed algorithm scaling to billions of edges
(Distributed) Triangle Counting

Sequential (Latapy 2006)
- orient edges
- iterate over all edges and intersect outgoing neighborhoods
(Distributed) Triangle Counting

Sequential (Latapy 2006)
- orient edges
- iterate over all edges and intersect outgoing neighborhoods
(Distributed) Triangle Counting

Sequential (Latapy 2006)
- orient edges
- iterate over all edges and intersect outgoing neighborhoods
(Distributed) Triangle Counting

**Sequential** (Latapy 2006)
- **orient** edges
- iterate over all edges and **intersect** outgoing neighborhoods

**Distributed Memory** (Arifuzzaman et al. 2015)
- **1D partitioning** of the vertex set
- send neighborhoods to adjacents PEs
(Distributed) Triangle Counting

Sequential (Latapy 2006)
- orient edges
- iterate over all edges and intersect outgoing neighborhoods

Distributed Memory (Arifuzzaman et al. 2015)
- 1D partitioning of the vertex set
- send neighborhoods to adjacents PEs
(Distributed) Triangle Counting

**Sequential** (Latapy 2006)
- orient edges
- iterate over all edges and intersect outgoing neighborhoods

**Distributed Memory** (Arifuzzaman et al. 2015)
- 1D partitioning of the vertex set
- send neighborhoods to adjacents PEs

**But**
- superlinear communication volume
- volume dependent on neighborhood size
- irregular communication pattern
Algorithmic Building Blocks

Point-to-point model

\[ \alpha + \beta \ell \]
Algorithmic Building Blocks

- Point-to-point model
  - Startup overhead
  - Bandwidth
  - Message length
  \[ \alpha + \beta l \]
Algorithmic Building Blocks

Point-to-point model

\[ \alpha + \beta \ell \]

Startup overhead

Message length

Bandwidth

message buffering
indirect messaging
exploiting locality
Algorithmic Building Blocks

Point-to-point model

- Startup overhead
  - limit buffer to local input size
  - flush via non-blocking send on overflow

- Bandwidth
- Message length
- Bandwidth
- Message length

- Message buffering
- indirect messaging
- exploiting locality
Algorithmic Building Blocks

Point-to-point model

\[ \alpha + \beta \ell \]

- Startup overhead
- Bandwidth
- Message length

- limit buffer to local input size
- flush via non-blocking send on overflow

- message buffering
- indirect messaging
- exploiting locality
Algorithmic Building Blocks

Point-to-point model

- Startup overhead
- Bandwidth
- Message length

- limit buffer to local input
- flush via non-blocking send on overflow
- message buffering
- indirect messaging
- exploiting locality
Algorithmic Building Blocks

Point-to-point model

- Startup overhead
- Bandwidth
- Message length
- Message length

- limit buffer to local input size
- flush via non-blocking send on overflow
- message buffering
- indirect messaging
- exploiting locality

\[
\alpha + \beta \ell
\]
Algorithmic Building Blocks

Point-to-point model

- Startp up overhead
  - limit buffer to local input size
  - flush via non-blocking send on overflow

- Bandwidth
  - Message length
    - exploiting locality

- Message length
  - indirect messaging

- Bandwidth
  - point-to-point model
  - Startup overhead
  - Bandwidth
  - Message length
    - exploiting locality

- Algorithmic Building Blocks

α + βℓ

- limit buffer to local input size
- flush via non-blocking send on overflow

- Message buffering
- indirect messaging
Algorithmic Building Blocks

Point-to-point model

- Startup overhead
  - limit buffer to local input size
  - flush via non-blocking send on overflow

- Bandwidth
- Message length
- Bandwidth
- Message length

- indirect messaging
- exploiting locality
- message buffering
Algorithmic Building Blocks

Point-to-point model

- Startup overhead
  - limit buffer to local input size
  - flush via non-blocking send on overflow
- Message length
  - type-1
  - type-2
  - type-3
- Bandwidth
- Message length
- message buffering
- indirect messaging
- exploiting locality

\[ \alpha + \beta \ell \]
Algorithmic Building Blocks

Point-to-point model

- Startup overhead
  - limit buffer to local input size
  - flush via non-blocking send on overflow

- Message length
  - type-3

- Message length
  - exploiting locality

- Bandwidth

- Message length
  - indirect messaging

- Message length
  - message buffering

- Alpha + Beta \( l \)
Experimental Setup

Competitors based on MPI
- HavoqGT by Pearce et al. (HPEC 2019)
- TriC by Ghosh and Halappanavar (HPEC 2020)
- ours (four variants)

Instances
- Synthetic instances from KaGen generator (weak scaling)
  - RGG-2D, RHG, GNM, RMAT
  - up to $2^{37}$ edges
- Large real-world instances (strong scaling)

Hardware
- evaluated using up to 32 768 ($= 2^{15}$) cores of SuperMUC-NG thin nodes
Evaluation

\[ \alpha + \beta \ell \]

Startup overhead
- message buffering
  - limit buffer to local input size
  - flush via \texttt{non-blocking} send on overflow
- indirect messaging
Evaluation

Startup overhead
message buffering
- limit buffer to local input size
- flush via non-blocking send on overflow
indirect messaging

\[ \alpha + \beta \ell \]

Evaluation chart:
- twitter
- \( \text{ours} \)
- \( \text{ours} + \text{indirection} \)
- HAVOQGT
- TRIC

Graph:
- Time (s) on y-axis
- Cores on x-axis
- Log scale for time (s)
Evaluation

\[ \alpha + \beta \ell \]

Startup overhead
- message buffering
  - limit buffer to local input size
  - flush via non-blocking send on overflow
- indirect messaging

indirect messaging

\begin{figure}
\centering
\includegraphics[width=\textwidth]{chart.png}
\caption{Evaluation of triangle counting algorithms.}
\end{figure}
Evaluation

\[ \alpha + \beta \ell \]

Message length

exploiting locality

\text{type-3}

RGG2D\((2^{18})\)

RHG\((2^{18})\)

Evaluation

- ours
- ours + contraction
- ours + contraction + indirection
- HAVOQGT
- TRIC
Evaluation

\[ \alpha + \beta \ell \]

Message length
exploiting locality

\[ \ell \]
type-3

Evaluation

\begin{align*}
\text{time (s)} & \\
\text{volume} & \\
\text{cores} & \\
\end{align*}

\begin{align*}
\text{RGG2D}(2^{18}) & \\
\text{RHG}(2^{18}) & \\
\end{align*}

- ours
- ours + contraction
- ours + contraction + indirection
- HAVOQGT
- TRIC
Conclusion

Our contributions:
Massively Scalable Distributed Triangle Counting Algorithms

- linear memory requirements using non-blocking sparse all-to-all algorithm
- reduce startup overhead by message aggregation and indirect communication
- reduce communication by local computations
- up to $18 \times$ better performance
- scalable to up to at least 32 768 cores

Future work:
- engineered hybrid implementation
- use communication primitives as foundation for general purpose graph processing framework
Appendix

Full Results
Full Weak Scaling Experiments

**Graphs:**
- **RGG2D($2^{18}$)**
- **RHG($2^{18}$)**
- **GNM($2^{16}$, $2^{20}$)**
- **RMAT($2^{16}$)**

**Axes:**
- **Time (s)**
- **Sent Messages**
- **Volume**

**Legend:**
- DITRIC
- DITRIC$^2$
- CETRIC
- CETRIC$^2$
- HAVOQGT
- TriC

**Scenarios:**
- $2^{11}$
- $2^{14}$
- $2^{17}$
- $2^{20}$
- $2^{23}$

**Additional Information:**
- May 18, 2023
- Tim Niklas Uhl: Engineering a Distributed-Memory Triangle Counting Algorithm
- Institute of Theoretical Informatics, Algorithm Engineering Group
Full Strong Scaling Experiments

Twitter

Webbase-2001

Live-journal

USA

Friendster

UK-2007-05

Orkut

Europe

Time (s)

DITRIC

DITRIC²

CETRIC

CETRIC²

HAVOQGT

TRIC
Phase Times

friendster

webbase-2001

live-journal

preprocessing
contraction
local phase
global phase
Hybrid Implementation

![Graphs showing the relationship between time, local time, and communication volume with varying numbers of cores and threads. The y-axis represents time in seconds, and the x-axis represents the number of cores which is equal to MPI ranks times threads.]

threads
- 1
- 3
- 6
- 12
- 24
- 48