# Naiad and Timely Dataflow [1]

## A theory first compute systems

2026-10-26

William B. Sørensen

# Contents

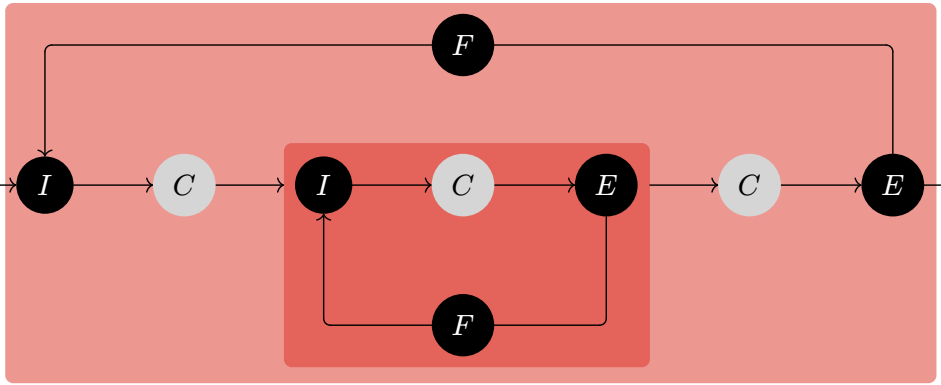

Timely dataflow [1], [2] ..... 3
What is Timely Dataflow? ..... 4
Graph structure ..... 5
Logical clock ..... 6
The compute model ..... 8
Alternative compute graphs ..... 9
Naiad ..... 10
What is Naiad ..... 11
Distributed compute concerns ..... 12
Optimizations ..... 13
Micro-Stragglers ..... 14
Performance and results ..... 15
Naiad today ..... 18
Contributions to the greater litearature ..... 19
Bibliography ..... 20

# Timely dataflow [1], [2]

# What is Timely Dataflow?

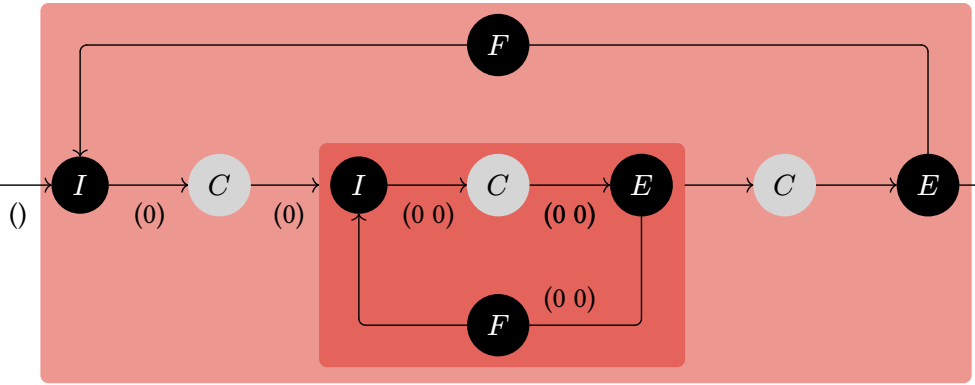
- A structured graph format,
- a logical clock and
- a compute model

# Graph structure

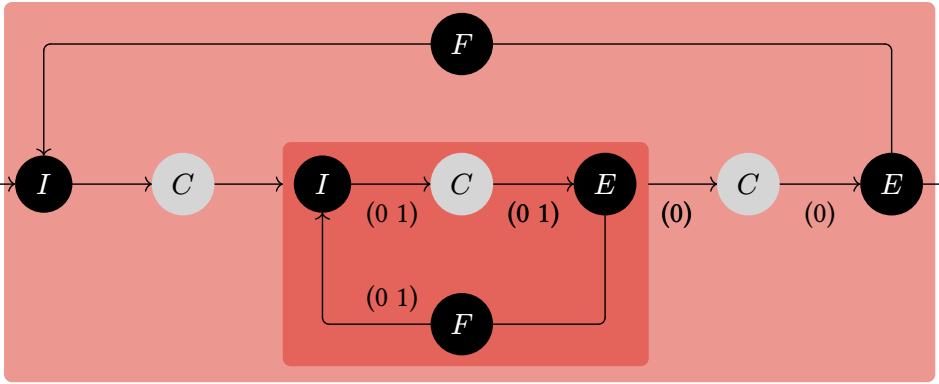



- Compute nodes $C$
- Loop context, bracketed by $I$ and $E$ nodes, where the only backlink is through an $F$ node
- Data flows through ingress nodes to egress nodes

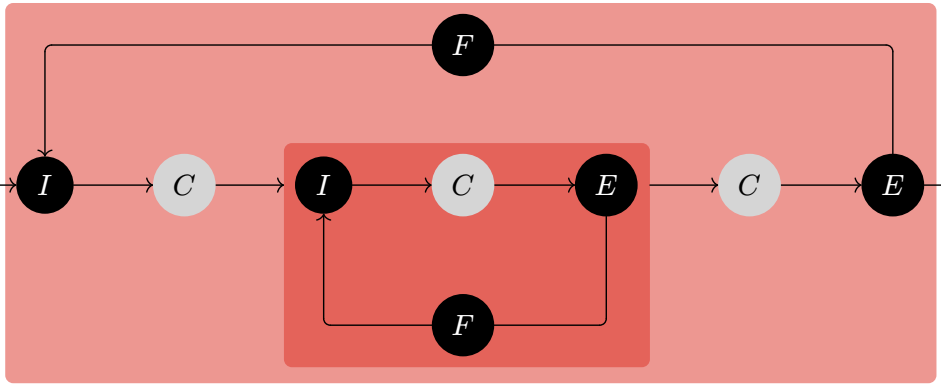
# Logical clock



# Logical clock (ii)



# The compute model




- Compute in *verticies*
- Flow through *edges*
- Computation can be done eagarly as long as effects are observed in timestamp order

# Alternative compute graphs



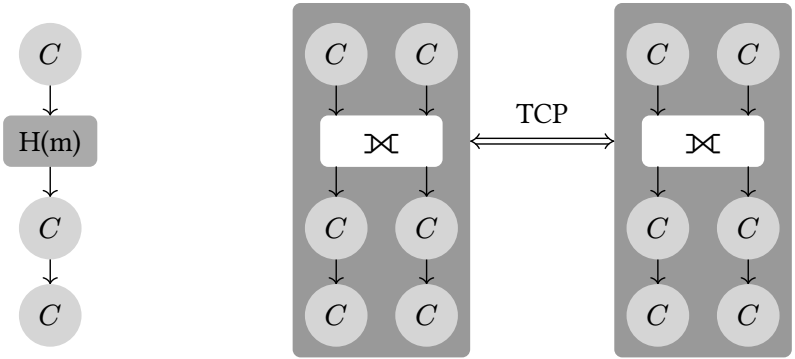
- MapReduce [3] has a fixed data-flow, so latencies multiply
- Dryad [4] does not support cycles
- CEIL [5] *dynamically* creates an acyclic control graph

# Naiad

# What is Naiad

- Naiad is a distributed implementation of timely data-flow
- Naiad is a multi-level distributed compute engine
  - LINQ
  - SQL
  - Low-level Naiad compute engine
  - MapReduce [3]
  - PowerGraph GAS [6]
- Naiad has a key focus on both latency and throughput
- Naiad has a set of optimization on top of this, key concern is *micro-stragglers*

# Distributed compute concerns




- Naiad has two levels of graph: physical and logical.
- Naiad relies on data parallelism edges can partition data across workers.
- Naiad can set the hash function to compute reduce / group-by operations.

# Optimizations

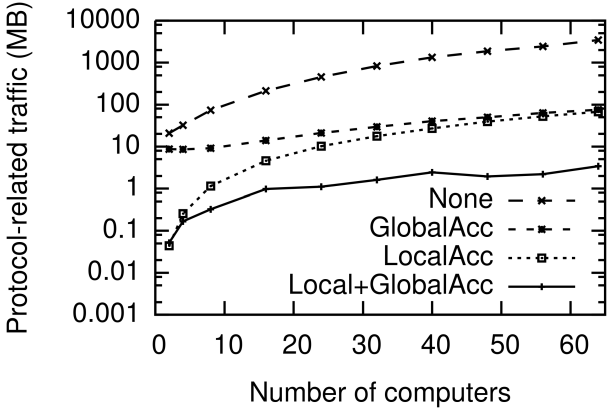


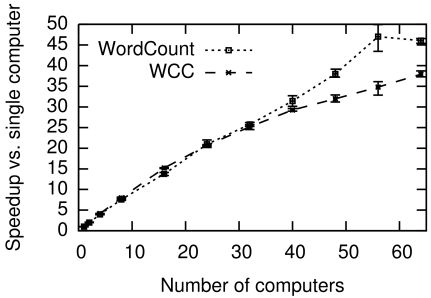

Figure 7 (c): from [1]

- Global Accumulation using a (pointstamp frontier)
- Local coalessing

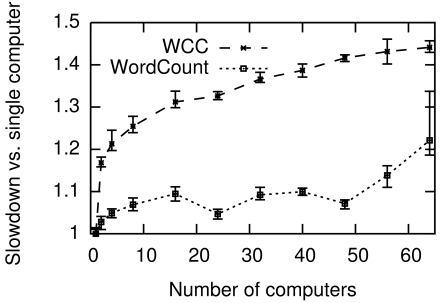
# Micro-Stragglers

- **Network**
  - Highly bursty
  - UDP
  - Tuned CC
  - DCTCP [7]
- **Windows clock granularity**
  - Spinlocks
- **.NET GC**
  - Spans
  - Low Frequency

# Performance and results




(d) Strong "Amdahl" scaling




(e) Weak "Gustafson" scaling

Figure 7: from [1]

- Poorish scaling
- Embarrassingly parallel and highly iterative workloads

# Performance and results (ii)

| Algo [s] | PDW | DryadLINQ | SHS | Naiad |
|---|---|---|---|---|
| PageRank | 156,982 | 68,791 | 836,455 | 4,656 |
| SCC | 7,306 | 6,294 | 15,903 | 729 |
| WCC | 214,479 | 160,168 | 26,210 | 268 |
| ASP | 671,142 | 749,016 | 2,381,278 | 1,131 |

# Performance and results (iii)

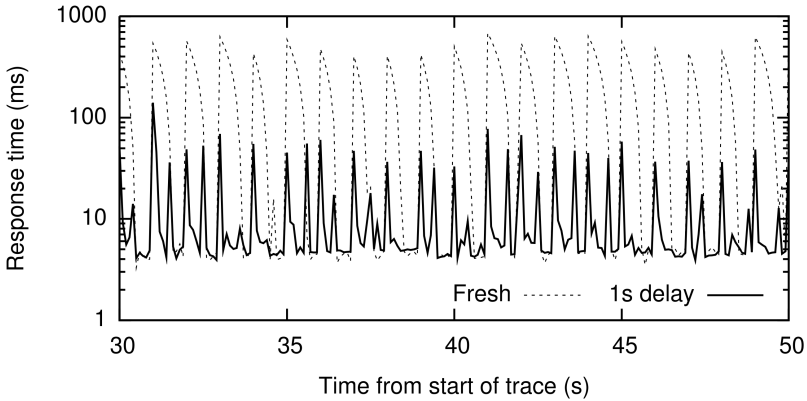



Figure 8: from [1]

- $k$-exposure 185kTw/s $\longrightarrow$ 483kWTw/s
- Delaying helpes response time

# Naiad today

# Contributions to the greater litearature

**Key contributions**

- Approach for timestamping and productivity [8], [9].
- Iterative graph computation [9], [10], [11].

**General contributions**

- Often a highlight stream implementation next to Spark [8], [12], [13], [14], [15], [16]
- Check-pointing [11], [14], [15].
- Micro-Straggler optimizations [14].

# Bibliography


[1] D. G. Murray, F. McSherry, R. Isaacs, M. Isard, P. Barham, and M. Abadi, "Naiad: a timely dataflow system," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, in SOSP '13. Farminton, Pennsylvania: Association for Computing Machinery, 2013, pp. 439–455. doi: 10.1145/2517349.2522738.

[2] M. Abadi, F. McSherry, D. G. Murray, and T. L. Rodeheffer, "Formal Analysis of a Distributed Algorithm for Tracking Progress," in *Formal Techniques for Distributed Systems*, D. Beyer and M. Boreale, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 5–19.

[3] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, San Francisco, CA, 2004, pp. 137–150.

[4] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed Data-parallel Programs from Sequential Building Blocks," in *Proceedings of the 2007 Eurosys Conference*, Association for Computing Machinery, Inc., Mar. 2007. [Online]. Available: https://www.microsoft.com/en-us/research/publication/dryad-distributed-data-parallel-programs-from-sequential-building-blocks/

[5] D. G. Murray, M. Schwarzkopf, C. Smowton, S. Smith, A. Madhavapeddy, and S. Hand, "CIEL: a universal execution engine for distributed data-flow computing," in *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, in NSDI'11. Boston, MA: USENIX Association, 2011, pp. 113–126.

[6] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin, "PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs," in *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, Hollywood, CA: USENIX Association, Oct. 2012, pp. 17–30. [Online]. Available: https://www.usenix.org/conference/osdi12/technical-sessions/presentation/gonzalez

[7] M. Alizadeh *et al.*, "Data center TCP (DCTCP)," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 63–74, Aug. 2010, doi: 10.1145/1851275.1851192.

[8] M. Fragkoulis, P. Carbone, V. Kalavri, and A. Katsifodimos, "A survey on the evolution of stream processing systems," *The VLDB Journal*, vol. 33, no. 2, pp. 507–541, Nov. 2023, doi: 10.1007/s00778-023-00819-8.

[9] M. Abadi *et al.*, "TensorFlow: a system for large-scale machine learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, in OSDI'16. Savannah, GA, USA: USENIX Association, 2016, pp. 265–283.

[10] J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica, "GraphX: graph processing in a distributed dataflow framework," in *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, in OSDI'14. Broomfield, CO: USENIX Association, 2014, pp. 599–613.

[11] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica, "Discretized streams: fault-tolerant streaming computation at scale," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, in SOSP '13. Farminton, Pennsylvania: Association for Computing Machinery, 2013, pp. 423–438. doi: 10.1145/2517349.2522737.

[12] G. Sheng *et al.*, "HybridFlow: A Flexible and Efficient RLHF Framework," in *Proceedings of the Twentieth European Conference on Computer Systems*, in EuroSys '25. Rotterdam, Netherlands: Association for Computing Machinery, 2025, pp. 1279–1297. doi: 10.1145/3689031.3696075.

[13] A. Margara, G. Cugola, N. Felicioni, and S. Cilloni, "A Model and Survey of Distributed Data-Intensive Systems," *ACM Comput. Surv.*, vol. 56, no. 1, Aug. 2023, doi: 10.1145/3604801.

[14] G. Theodorakis, F. Kounelis, P. Pietzuch, and H. Pirk, "Scabbard: single-node fault-tolerant stream processing," *Proc. VLDB Endow.*, vol. 15, no. 2, pp. 361–374, Oct. 2021, doi: 10.14778/3489496.3489515.

[15] M. Li *et al.*, "Scaling distributed machine learning with the parameter server," in *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, in OSDI'14. Broomfield, CO: USENIX Association, 2014, pp. 583–598.

[16] P. Moritz *et al.*, "Ray: a distributed framework for emerging AI applications," in *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, in OSDI'18. Carlsbad, CA, USA: USENIX Association, 2018, pp. 561–577.