UNIVERSITY OF CAMBRIDGE

Naiad: A Timely Dataflow System

Indigo Orton – R244

Motivation

- High throughput
- Low latency
- Interactive querying

Example – Analytics dashboard

- Constant metric streams – *stream*
- Automated insights – *stream + batch*
- Interactive user queries – *interactive*

Details

Key idea

- Records traveling through a graph
- “Timely dataflow”
- Timestamps - *progressive record ids*
- Timestamps - *loop counters*

Graph model

- Graph based computation model
- Enable loops within graph
- Highly parallel stream processing

Data integrity

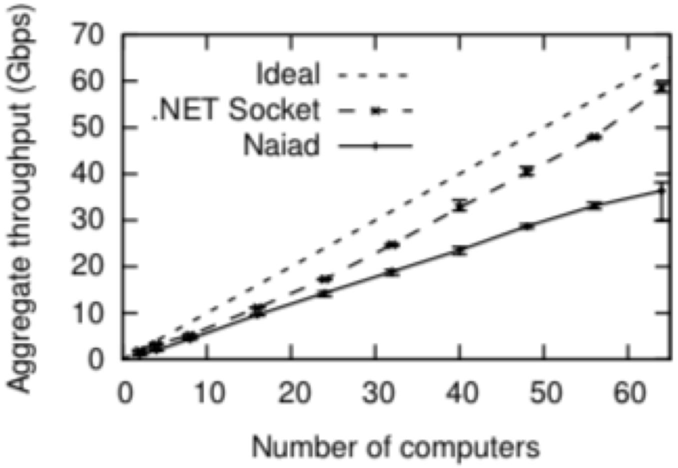
- Process records in epoch order
- Notifications to vertices – *i.e. flushing*
- Calculation of possible records

Limitation - Micro-stragglers

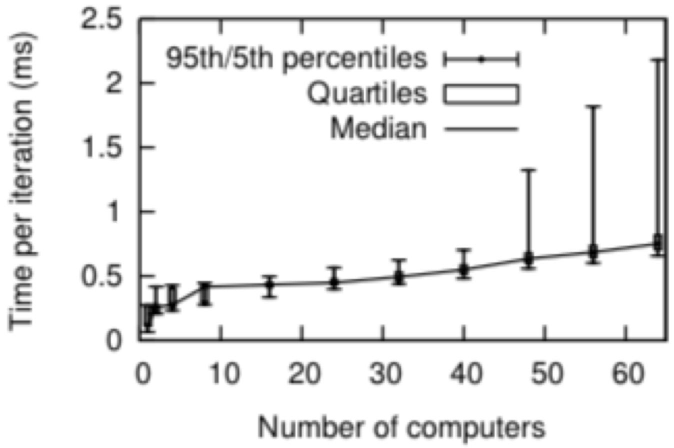
- Micro-stragglers – outsized performance impact
- Mutable shared state for low latency
- In-memory datasets

Results

Throughput



Latency



Twitter

Algorithm	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**

Context

- Vertex centric computation models - Pregel [2]
- TensorFlow [4] – uses timely dataflow in dynamic computation
- Straggler mitigation a higher priority in some systems – RDD [5], D-Streams [6] (based on RDD).
- Later systems decouple processing and coordination for faster cluster adaption – Drizzle [7]
- Updates to Naiad – last public commit in 2014 [3]
- Industry projects – Apache Flink™ [8]

Review

Encouraging highlights

- Graphs as a computational dependency model
- Modulization of computations
- Streaming, batch, and interactive support

Concerns

- Micro-stragglers – inability to mitigate
- Unsuitable for memory intensive computations
- Addressed via implementation optimisation
- Implementation approach and allocation of research resources
- Unnecessary complexity – timestamps/notifications

The paper

- Unnecessary complexity
  - Timestamps – *progressive ids*
  - Notifications – *flushing*
- Focus on implementation optimisations

The space – further discussion

- Nothing solves specifically for our target
- Collaboration between frameworks
- New framework that will not collaborate
- Generic protocol
- Jack of all trades, master of none

Conclusion

- Interesting model
- Modulization – global coordination
- Risks with micro-stragglers
- Unnecessary complexity
- Time spent on implementation optimisations
- Young field - or fundamentally unsolvable?

References

1. Murray, D. G., McSherry, F., Isaacs, R., Isard, M., 0001, P. B., & Abadi, M. (2013). Naiad - a timely dataflow system. *Sosp*, 439–455. http://doi.org/10.1145/2517349.2522738
2. Malewicz, G., Austern, M. H., Bik, A. J. C., Dehnert, J. C., Horn, I., Leiser, N., & Czajkowski, G. (2010). Pregel - a system for large-scale graph processing. *SIGMOD Conference*, 135. http://doi.org/10.1145/1807167.1807184
3. Naiad open source repository – Accessed 15/10/18 – https://github.com/MicrosoftResearch/Naiad
4. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., et al. (2016). TensorFlow - A System for Large-Scale Machine Learning. *CoRR*, *cs.DC*.
5. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauly, M., et al. (2012). Resilient Distributed Datasets - A Fault-Tolerant Abstraction for In-Memory Cluster Computing. *Nsdi*.
6. Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., & Stoica, I. (2013). Discretized streams - fault-tolerant streaming computation at scale. *Sosp*, 423–438. http://doi.org/10.1145/2517349.2522737
7. Venkataraman, S., Panda, A., Ousterhout, K., Armbrust, M., Ghodsi, A., Franklin, M. J., et al. (2017). Drizzle - Fast and Adaptable Stream Processing at Scale. *Sosp*, 374–389. http://doi.org/10.1145/3132747.3132750
8. Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., & Tzoumas, K. (2015). Apache Flink™ - Stream and Batch Processing in a Single Engine. *IEEE Data Eng. Bull.*