On The [Ir]relevance of Network Performance for Data Processing

Animesh Trivedi, Patrick Stuedi, Jonas Pfefferle, Radu Stoica, Bernard Metzler, Ioannis Koltsidas, Nikolas Ioannou

IBM Research, Zurich
How [Ir]relevant is the Network?

Making Sense of Performance in Data Analytics Frameworks
Kay Ousterhout*, Ryan Rastī*,† ⟨, Sylvia Ratnasamy*, Scott Shenker*,†, Byung-Gon Chun‡
*UC Berkeley, †ICSI, ‡VMware, ‡Seoul National University

Network optimizations can only reduce job completion time by a median of at most 2%. The network is not a bottleneck because much less data is sent over the network than is transferred to and from disk. As a result, network I/O is mostly irrelevant to overall performance, even on 1Gbps networks.
How [Ir]relevant is the Network?

Making Sense of Performance in Data Analytics Frameworks
Kay Ousterhout*, Ryan Rasti*†○, Sylvia Ratnasamy*, Scott Shenker*†, Byung-Gon Chun‡
*UC Berkeley, †ICSI, ○VMware, ‡Seoul National University

Network optimizations can only reduce job completion time by a median of at most 2%. The network is not a bottleneck because much less data is sent over the network than is transferred to and from disk. As a result, network I/O is mostly irrelevant to overall performance, even on 1Gbps networks.
How [Ir]relevant is the Network?

Runtime in secs

TeraSort  PageRank  SQL  WordCount  GroupBy
How [Ir]relevant is the Network?

Run time in secs

- TeraSort
- PageRank
- SQL
- WordCount
- GroupBy

- 1 Gbps
How [Ir]relevant is the Network?

Network IO is very relevant - up to 64%
How [Ir]relevant is the Network?

Network IO is very relevant - up to 64% ??
Is It Spark Specific?

![Bar chart showing runtime for different systems and data rates]

- **Flink-TS**
  - 1 Gbps: 725s
  - 10 Gbps: Not applicable
  - 40 Gbps: Not applicable

- **Flink-PR**
  - 1 Gbps: 120s
  - 10 Gbps: 150s
  - 40 Gbps: 200s

- **GraphLab**
  - 1 Gbps: 180s
  - 10 Gbps: 100s
  - 40 Gbps: 70s

- **Timely**
  - 1 Gbps: 300s
  - 10 Gbps: 150s
  - 40 Gbps: 50s
Spark TeraSort: The Shuffle Story

distributed sorting

- simple
- shuffle data is input data
- highest chance of improvements

input

output
Spark TeraSort: The Shuffle Story

Map tasks

Cores

Shuffle data

Reduce tasks

input

output
Spark TeraSort: The Shuffle Story

Map tasks

Cores

Shuffle data

Reduce tasks

input

output

reading in shuffle data
Spark TeraSort: The Shuffle Story

input

Map tasks

Cores

reading in shuffle data

sorting shuffle data

output

Reduce tasks

net CPU

net CPU

net CPU
Spark TeraSort: The Shuffle Story

The shuffle process involves reading in shuffle data, sorting shuffle data, and then combining these to achieve performance.
How Important is the Network?

Gains from the networks are shadowed by the high CPU footprint.
How Important is the Network?

Gains from the networks are shadowed by the high CPU footprint.
How Important is the Network?

Gains from the networks are shadowed by the high CPU footprint.

- 1 Gbps: 48%
- 10 Gbps: 92%
- 40 Gbps: 8%
- 100 Gbps*: 52%
How Important is the Network?

The gains from the networks are shadowed by the high CPU footprint.
How Important is the Network?

Network gains are shadowed by the CPU
What Exactly is the CPU Doing?

[Chart showing CPU activities with categories like Map, Reduce, Misc., Iterator, Serialization, Sorting, IO, JVM, Linux, and Spark]
What Exactly is the CPU Doing?

Spark

Map

Reduce

- Misc.
- Iterator
- Serialization
- Sorting
- IO
- JVM
- Linux
Overheads are spread across the entire stack - serialization, abstration, execution model etc.

Spark
The Balancing Act: CPU vs Network
The Balancing Act: CPU vs Network

I. Balance out the CPU with the network time

Sorting: $O(n\log(n))$
Network: $O(n)$

use smaller '$n'$
The Balancing Act: CPU vs Network

I. Balance out the CPU with the network time

![Graph showing runtime (secs) vs smaller partitions]
The Balancing Act: CPU vs Network

I. Balance out the CPU with the network time

![Graph showing the relationship between smaller partitions and runtime (seconds)]
The Balancing Act: CPU vs Network

I. Balance out the CPU with the network time
The Balancing Act: CPU vs Network

I. Balance out the CPU with the network time

II. Use more cores to scale up

if a single core cannot do 40 Gbps
then use more
The Balancing Act: CPU vs Network

I. Balance out the CPU with the network time

II. Use more cores to scale up

---

The 8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '16)
The Balancing Act: CPU vs Network

I. Balance out the CPU with the network time

II. Use more cores to scale up
The Balancing Act: CPU vs Network

I. Balance out the CPU with the network time

II. Use more cores to scale up

![Graph showing the relationship between number of cores and bandwidth](image)

- Ideal and measured bandwidths are compared for different numbers of cores.
- The graph illustrates the trade-off between increasing CPU cores and the network bandwidth required to maintain performance.

---

The 8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '16)
The Balancing Act: CPU vs Network

I. Balance out the CPU with the network time

II. Use more cores to scale up

![Graph showing runtime vs smaller partitions](image)

![Bar chart showing runtime for different numbers of cores](image)
The Balancing Act: CPU vs Network

I. Balance out the CPU with the network time

II. Use more cores to scale up

\[
\text{runtime} = 9 + \frac{260}{\text{cores}}
\]
The Balancing Act: CPU vs Network

I. Balance out the CPU with the network time

II. Use more cores to scale up

Classical techniques are ineffective
Conclusion

1. Faster networks (IO) are very relevant
   - as long as you have CPU cycles
   - differentiate between user vs framework CPU usage
Conclusion

1. Faster networks (IO) are very relevant
   - as long as you have CPU cycles
   - differentiate between user vs framework CPU usage

2. Framework's CPU usage is bad
   - CPU-network imbalance: sorting, serialization, volcano execution model, etc.
   - scalability (serial vs parallel components)
   - ineffective classical balancing techniques
Conclusion

1. Faster networks (IO) are very relevant
   - as long as you have CPU cycles
   - differentiate between user vs framework CPU usage

2. Framework's CPU usage is bad
   - CPU-network imbalance: sorting, serialization, volcano execution model, etc.
   - scalability (serial vs parallel components)
   - ineffective classical balancing techniques

3. Knowing today's usec-era IO and CPU hardware, how would you re-design modern data processing framework?