Spark Terasort and CRAIL

www.crail.io

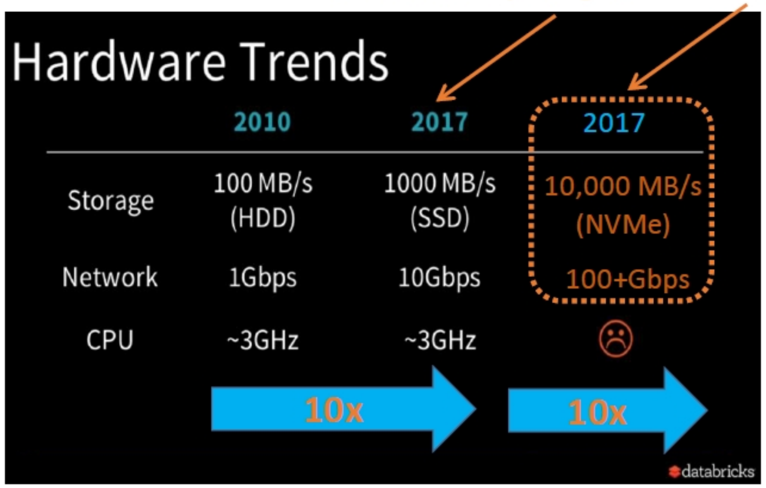
Peter Hofstee
IBM Research Austin

I/O Hardware Trends

Speed

community target

our target

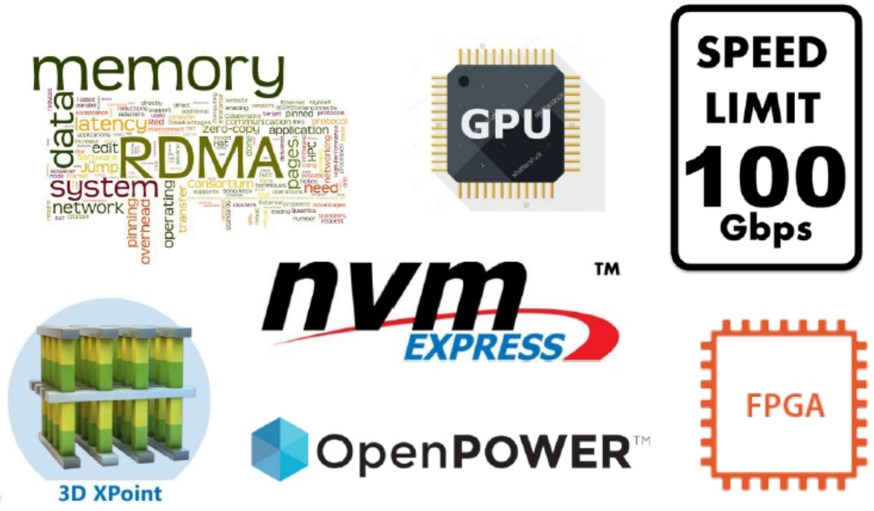


Hardware Trends

	2010	2017	2017
Storage	100 MB/s (HDD)	1000 MB/s (SSD)	10,000 MB/s (NVMe)
Network	1Gbps	10Gbps	100+Gbps
CPU	~3GHz	~3GHz	☹
	10x		10x

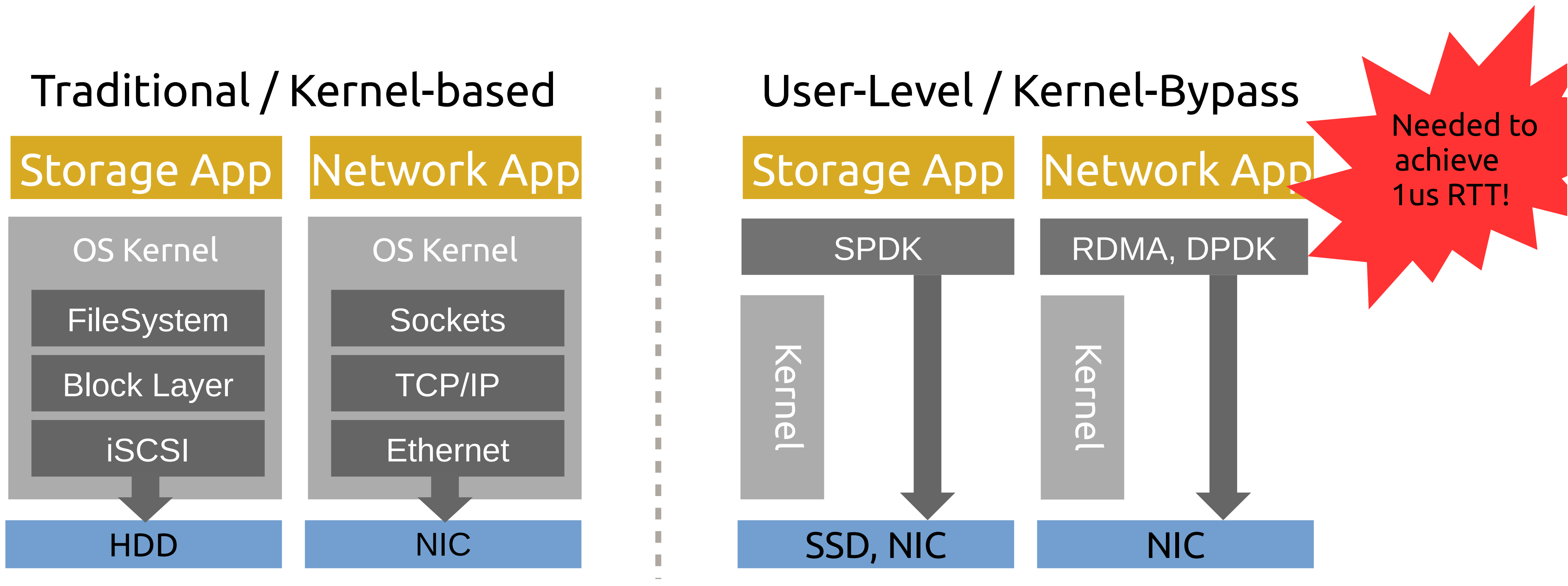
databricks

Diversity



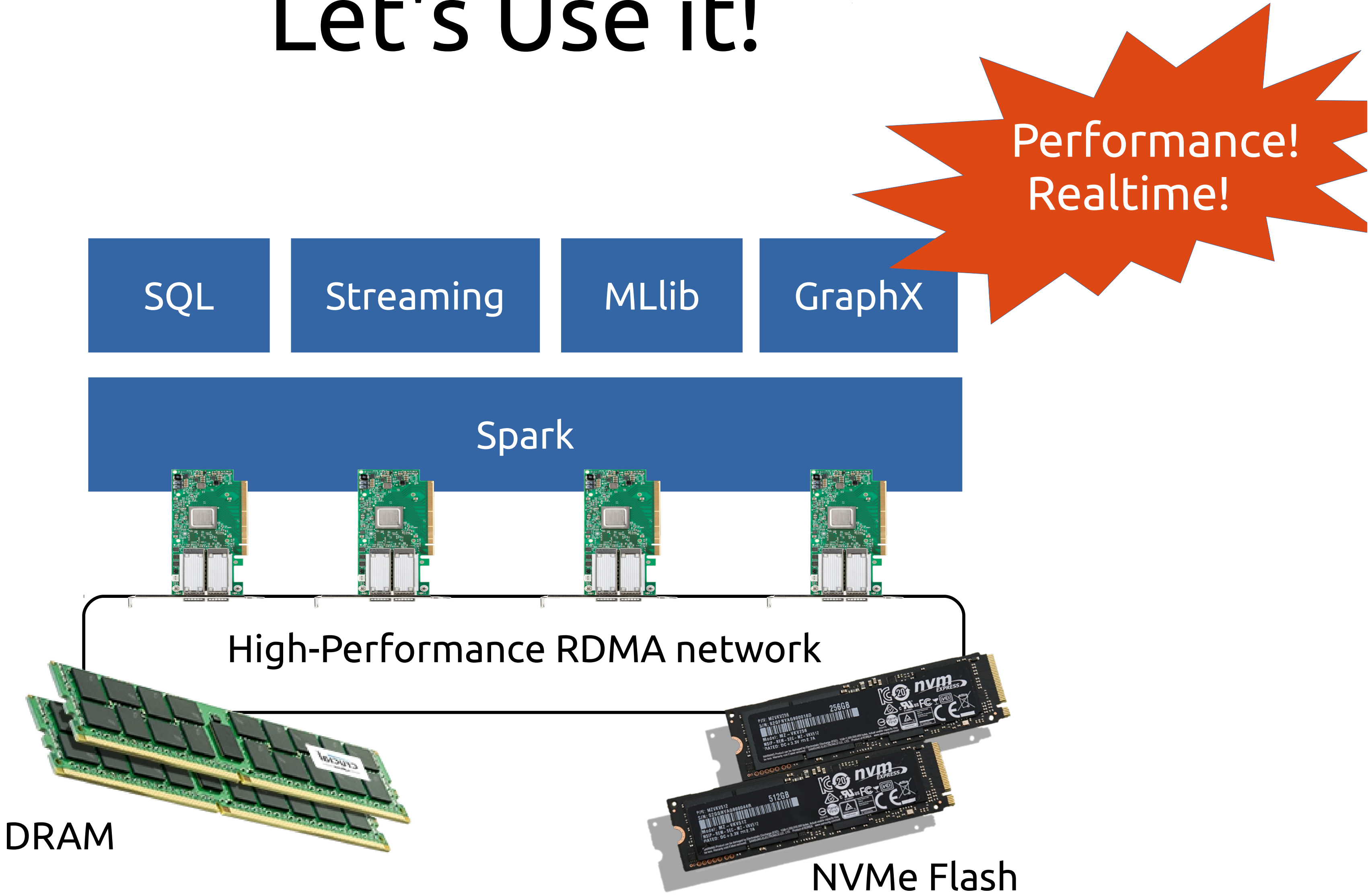
- Network interconnects have evolved from
  - Gbps bandwidth to 100Gbps
  - 100us delay to 1us delay
- Storage technology has evolved
  - Factor 100x-1000x

User-Level APIs

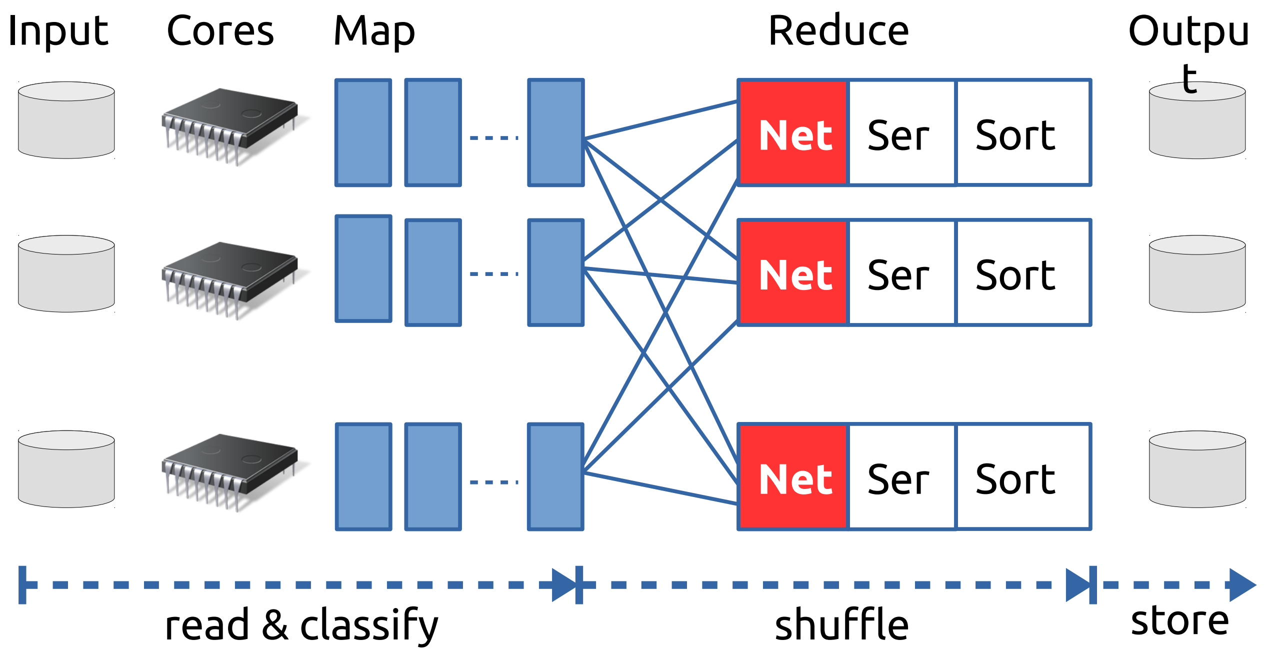


Modern APIs for Networking and Storage offer asynchronous non-blocking user-level access to hardware

Let's Use it!


Performance!
Realtime!
SQL
Streaming
MLlib
GraphX
Spark
High-Performance RDMA network
DRAM
NVMe Flash

Case Study: Sorting in Spark

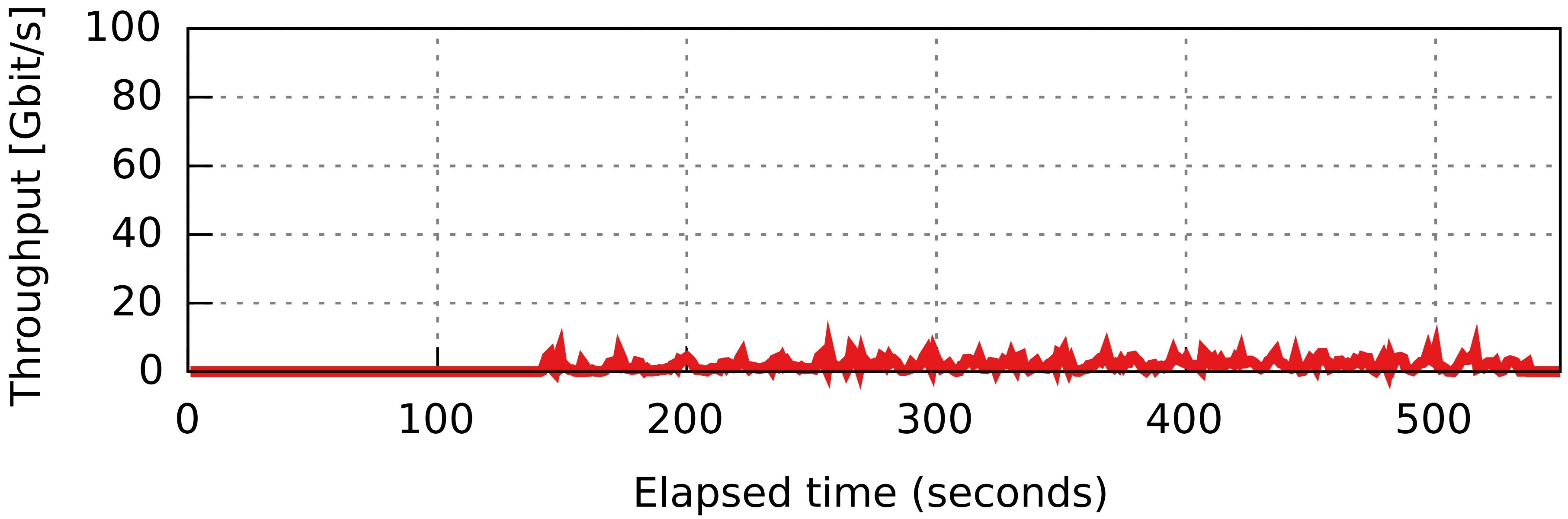


- Map task classify data into local files (typically absorbed by buffer cache)
- Reduce task fetch remote files over the network
- Sorting requires the entire data set to be shuffled over the network

Experiment Setup

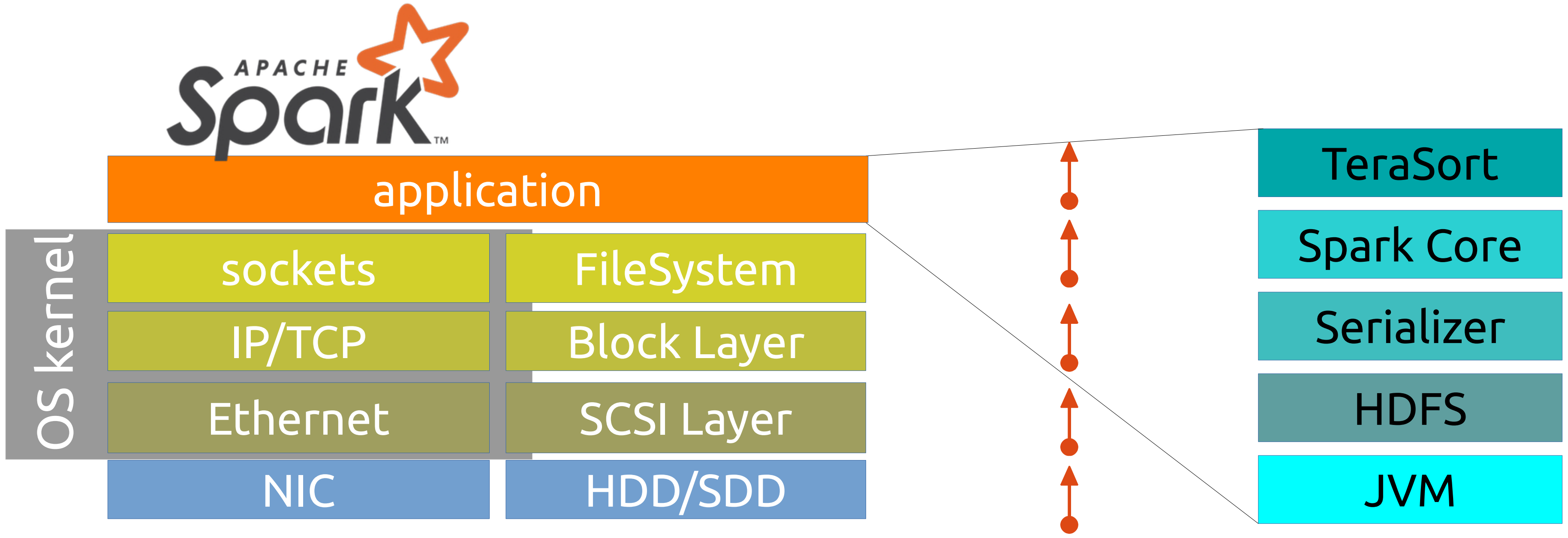
- Total data size: 12.8 TB
- Cluster size: 128 nodes
- Cluster hardware
  - DRAM: 512 GB DDR 4
  - Storage: 4x 1.2 TB NVMe SSD
  - Network: 100GbE Mellanox RDMA
- Software
  - Spark 2.0.0

How is the Network Used?



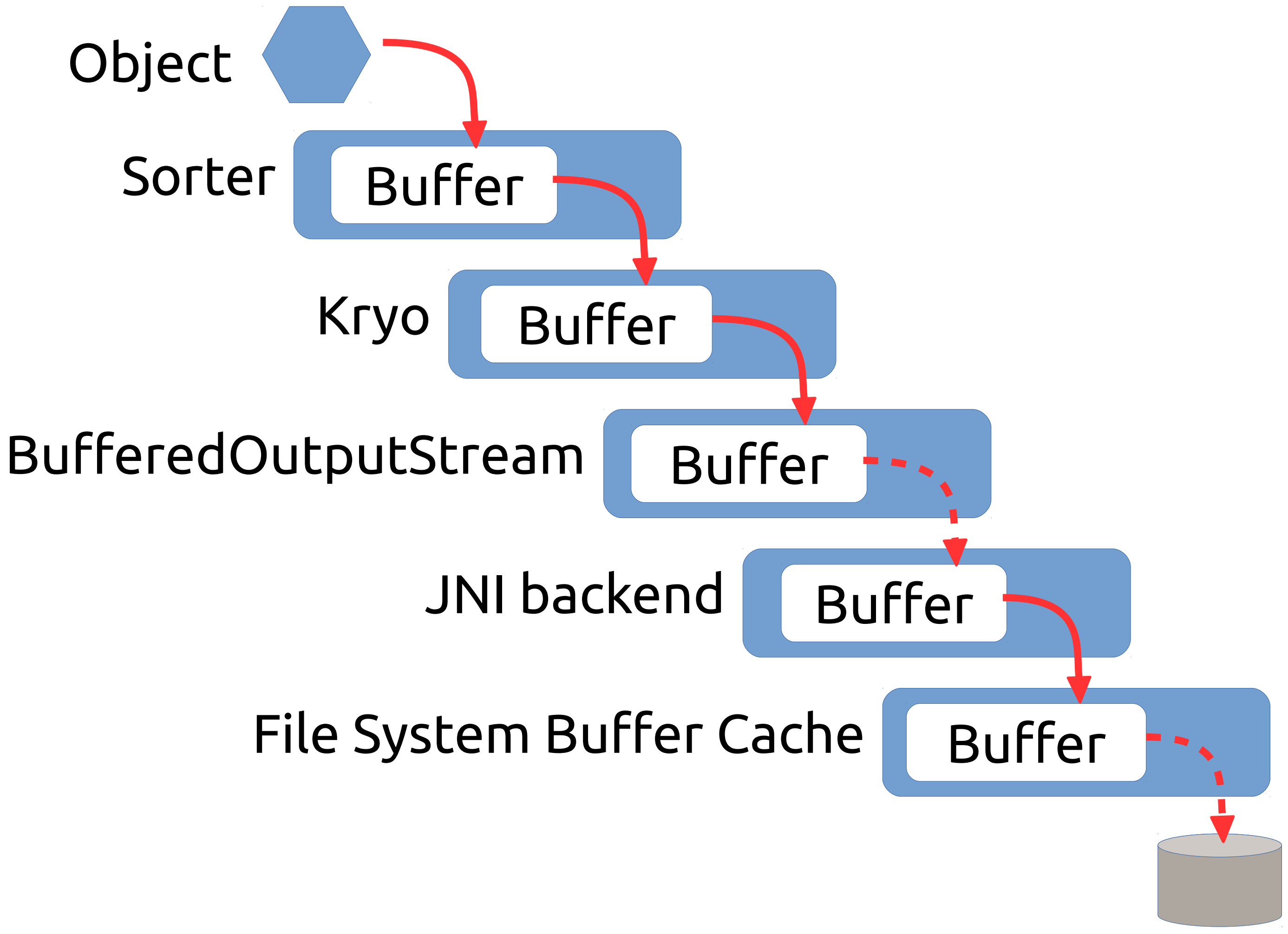
- Only 5-10 Gpbs of the network is being used

What is the Problem

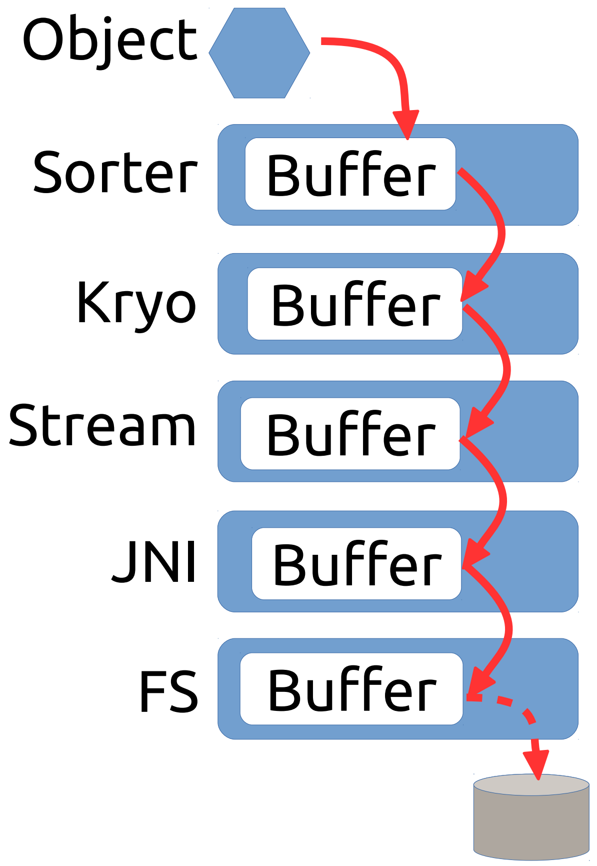


- Application use the legacy APIs
- Applications themselves are heavily layered!
- Overhead during local file system writing
- Overhead during network processing
  - Data copies, context switches, cache pollution, etc

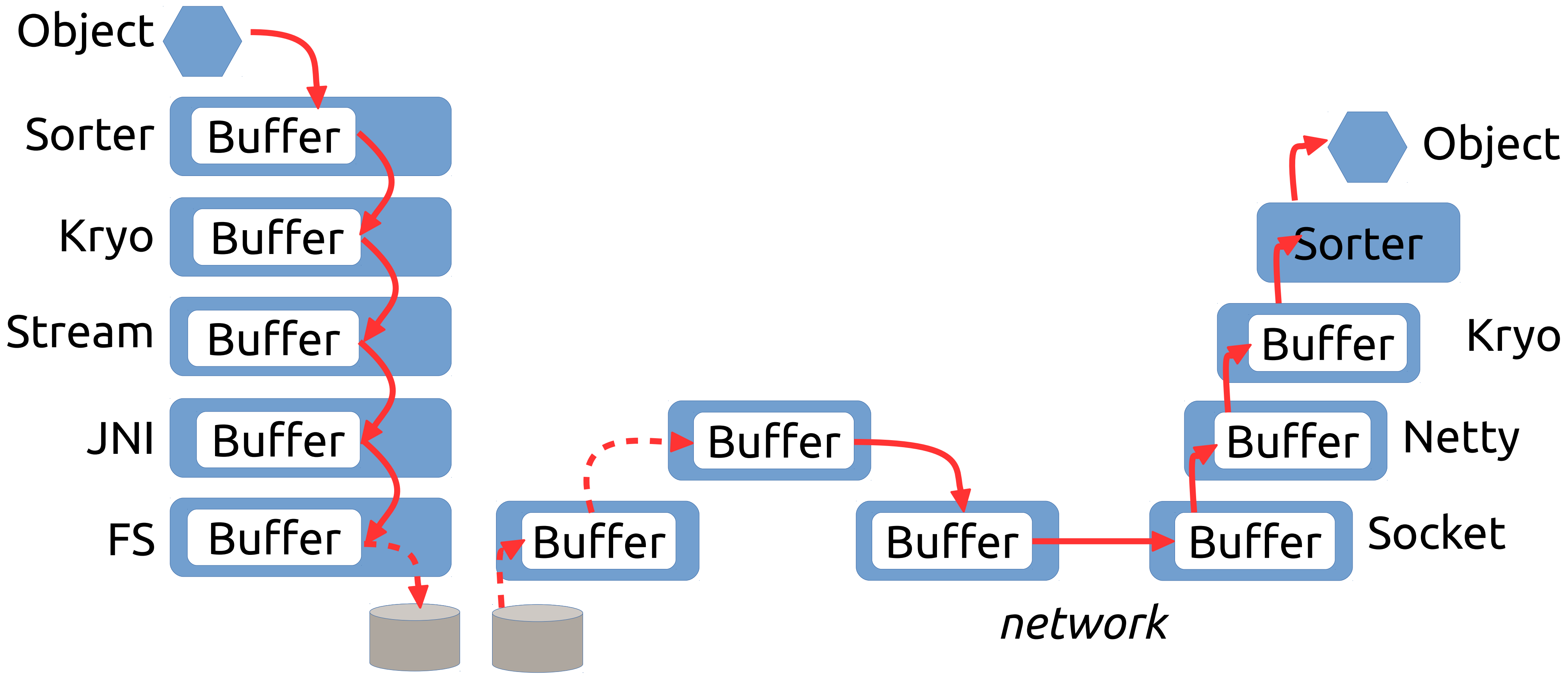
Example: Shuffle Writer (map)



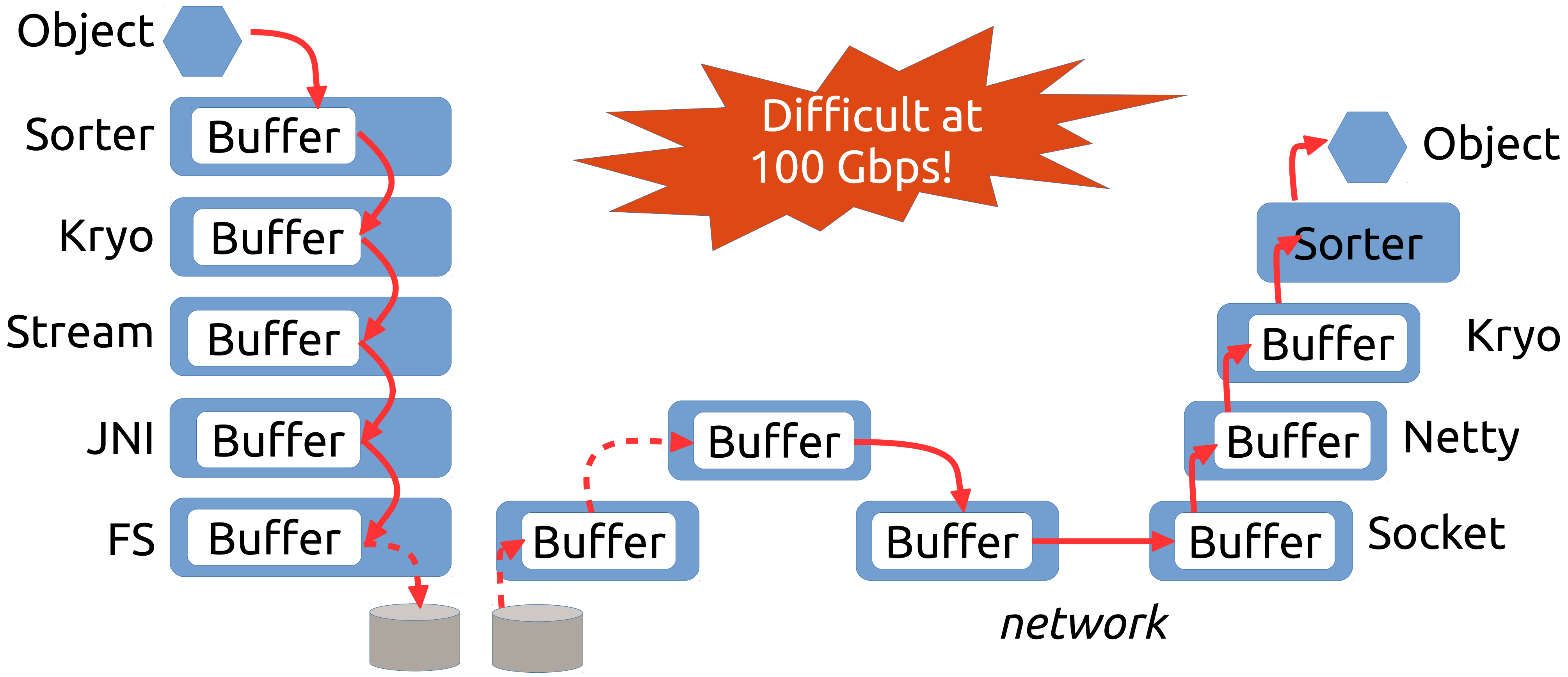
Example: Shuffle Writer (map)



Example: Shuffle Writer (map)



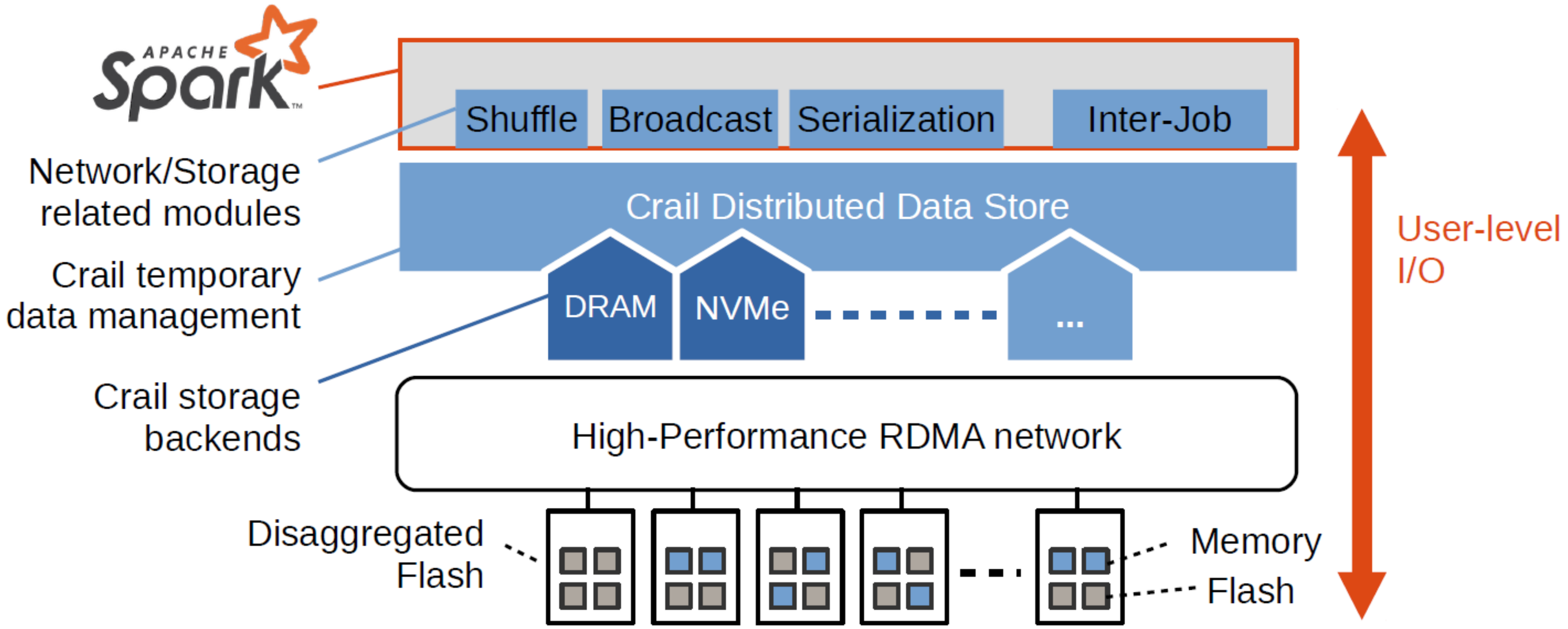
Example: Shuffle Writer (map)



How can we fix this...

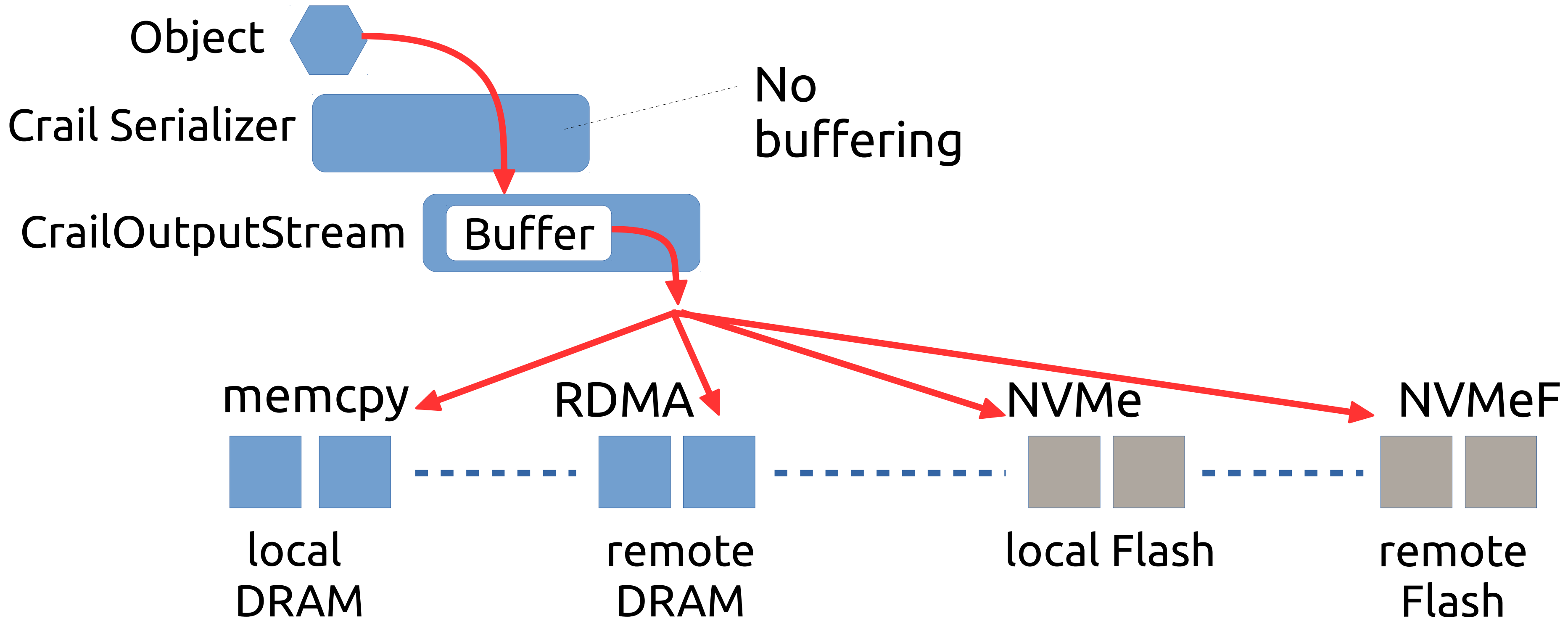
- Not just for shuffle
  - For broadcast, RDD transport, inter-job sharing, etc.
- Not just for RDMA and NVMe
  - For any future high-performance I/O hardware
- Not just for co-located compute/storage
  - Also for disaggregated storage, heterogeneous resource distribution, etc.
- Not just improve things
  - Make it perform at the hardware limit

The CRAIL Approach



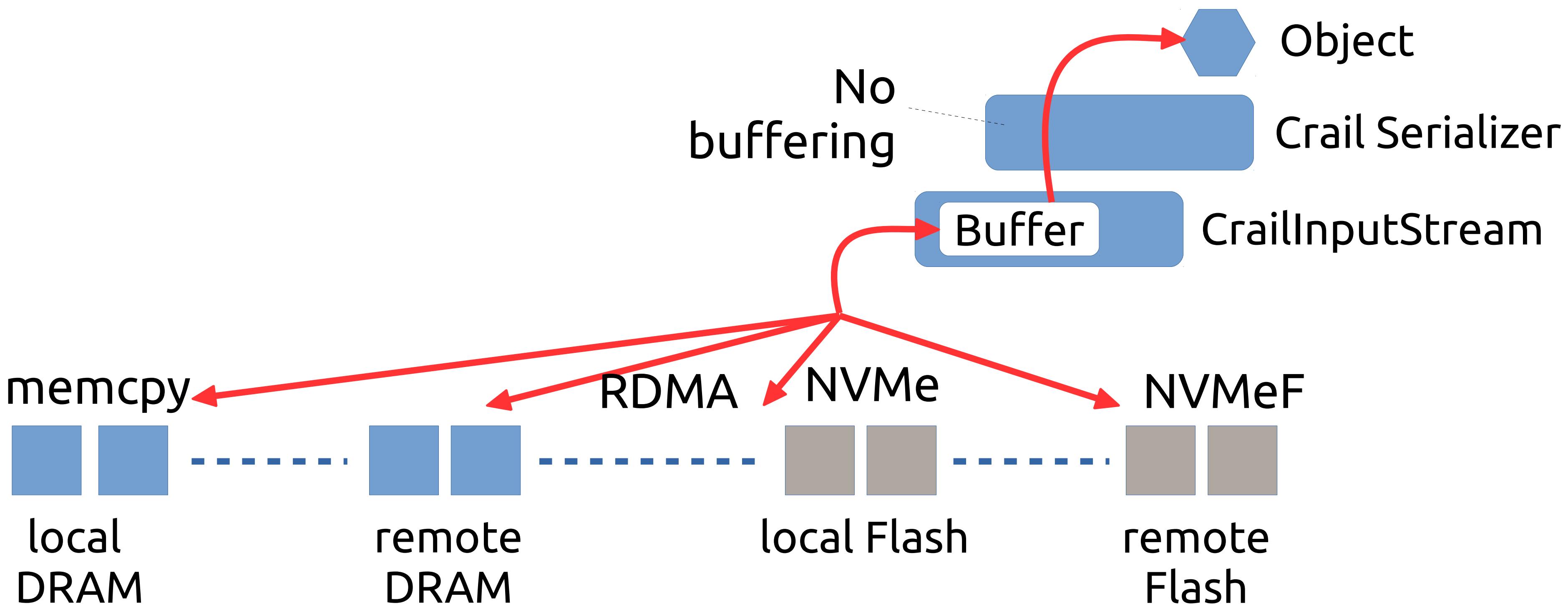
Re-think how I/O is handled in case of fast networking and storage hardware

Example: Crail Shuffle (map)



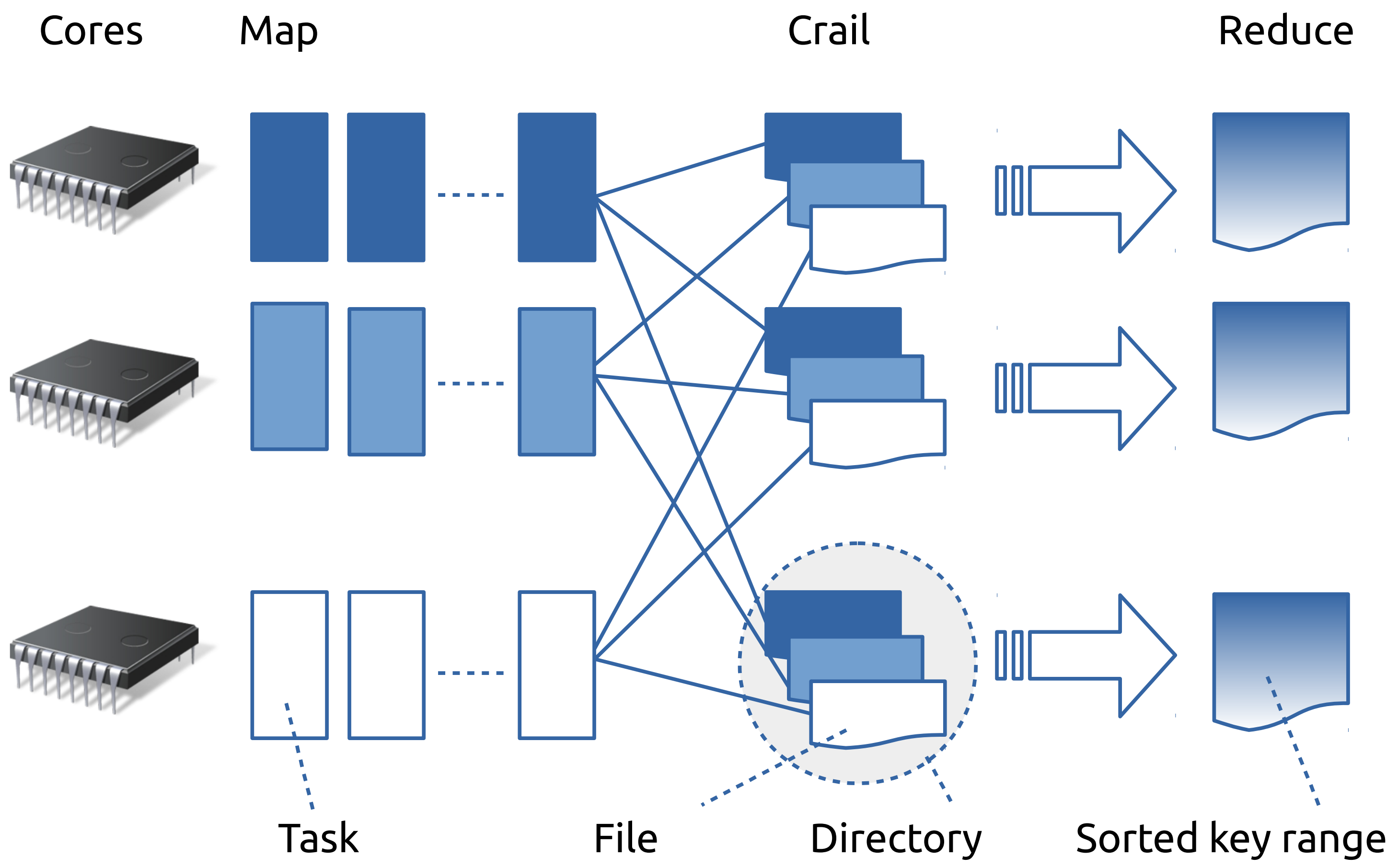
Higher-performing tiers are filled up across the cluster prior to using lower performing tiers

Example: Crail Shuffle (reduce)



Other Spark I/O operations such as broadcast, SQL join, etc., are implemented similarly

Crail Shuffle: File System Layout



Evaluation – Terasort

128 nodes OpenPOWER cluster

- 2 x IBM POWER8 10-core @ 2.9 GHz
- DRAM: 512GB DDR4
- 4 x 1.2 TB NVMe SSD
- 100GbE Mellanox ConnectX-4 EN (RoCE)
- Ubuntu 16.04 (kernel 4.4.0-31)
- Spark 2.0.2

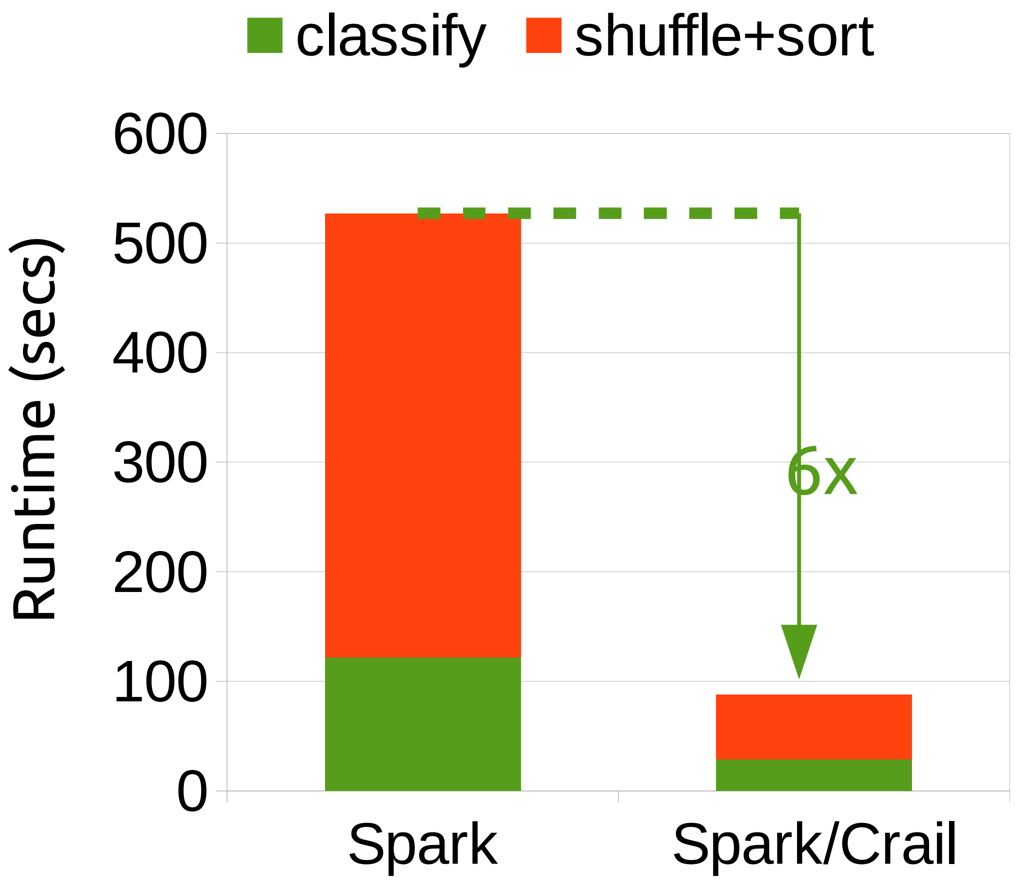
Evaluation – Terasort

128 nodes OpenPOWER cluster

- 2 x IBM POWER8 10-core @ 2.9 GHz
- DRAM: 512GB DDR4
- 4 x 1.2 TB NVMe SSD
- 100GbE Mellanox ConnectX-4 EN (RoCE)
- Ubuntu 16.04 (kernel 4.4.0-31)
- Spark 2.0.2

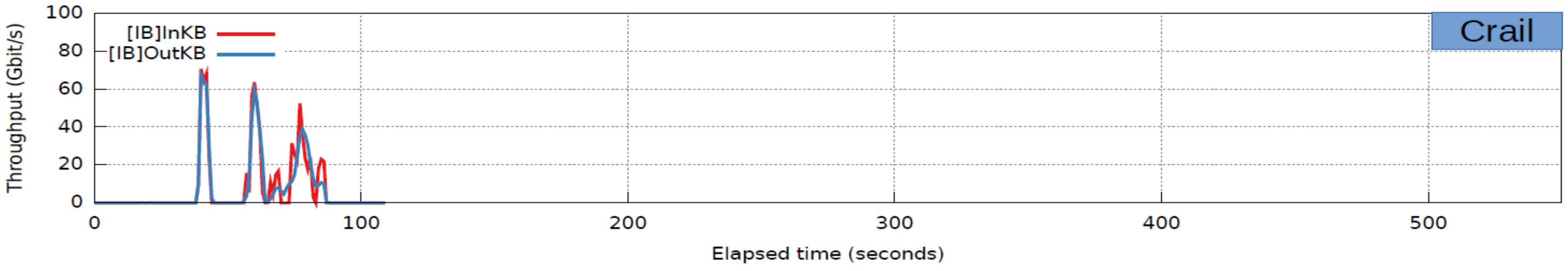
Performance gain: 6x

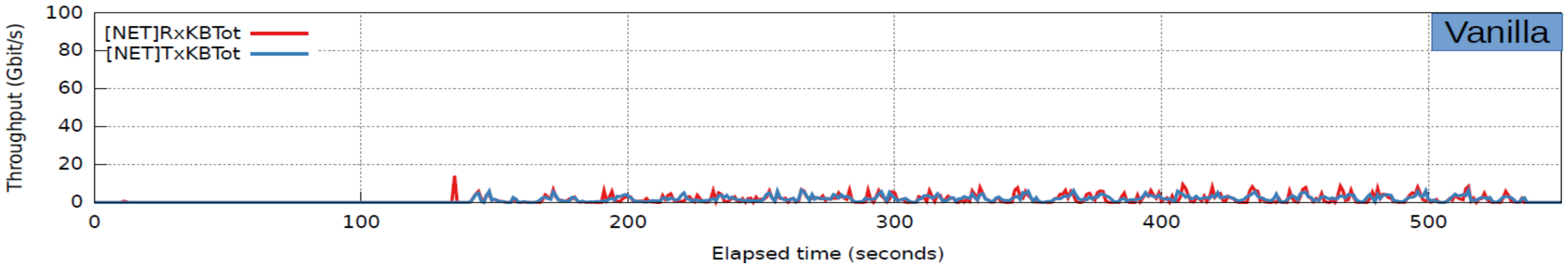
- Most gain from reduce phase:
  - Crail shuffler much faster than Spark build-in
  - Dramatically reduced CPU involvement
  - Dramatically improved network usage
- Map phase: all activity local
  - Still faster than vanilla Spark



12.8 TB data set, TeraSort

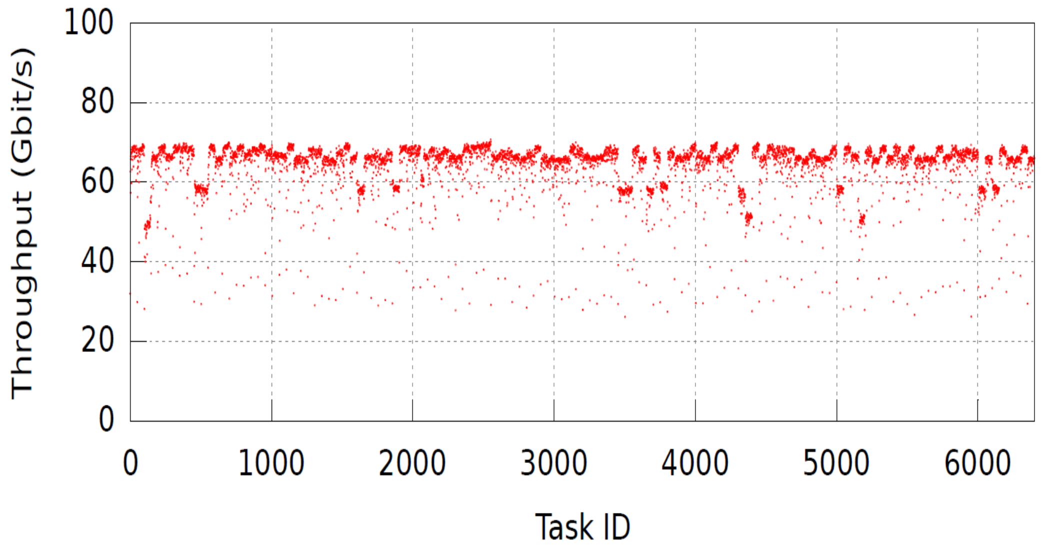
Evaluation – Network IO





- Vanilla Spark runs on 100GbE
- Spark/Crail runs on 100Gb RoCE/RDMA

- Vanilla Spark peaks at ~10Gb/s
- Spark/Crail shuffle delivers ~70Gb/s

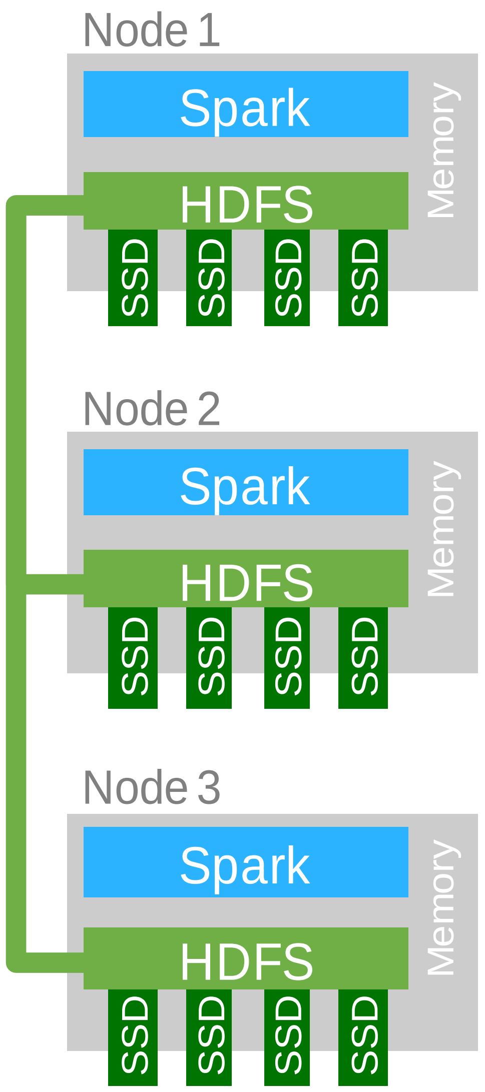


Sorting Comparison

	Spark + Crail	Spark 2.0.2	Winner 2014	Winner 2016
Size TB	12.8		100	
Time sec	98	527	1406	134
Cores	2560		6592	10240
Nodes	128		206	512
NW Gb/s	100		10	100
Rate TB/min	7.8	1.4	4.27	44.78
Rate/core GB/min	3.13	0.58	0.66	4.4

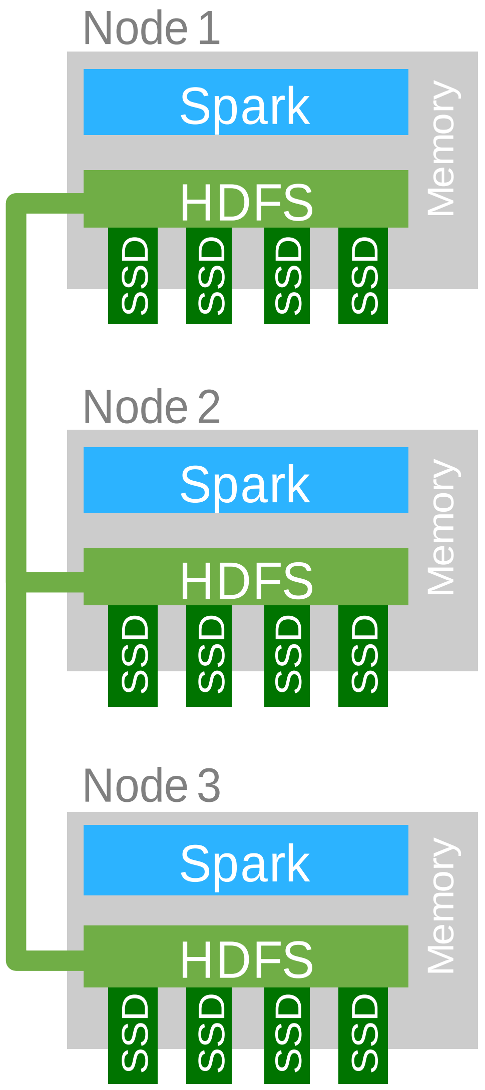
- Spark/Crail CPU efficiency is close to 2016 sorting benchmark winner: **3.13 vs. 4.4 GB/min/core**
- 2016 winner runs native C code!

Storage Disaggregation

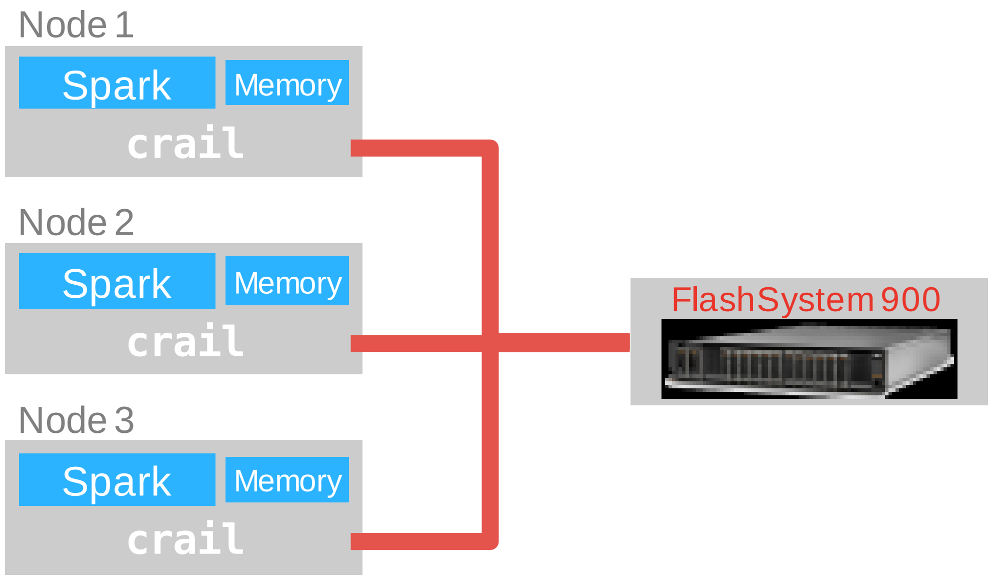


- Why disaggregation?
  - Independent scaling of compute and storage
  - Higher utilization due to less fragmentation
  - Easier maintenance
- Challenges:
  - Systems like Hadoop/Spark have been designed for local storage
  - But: new <u>fast networks</u> may permit storage disaggregation

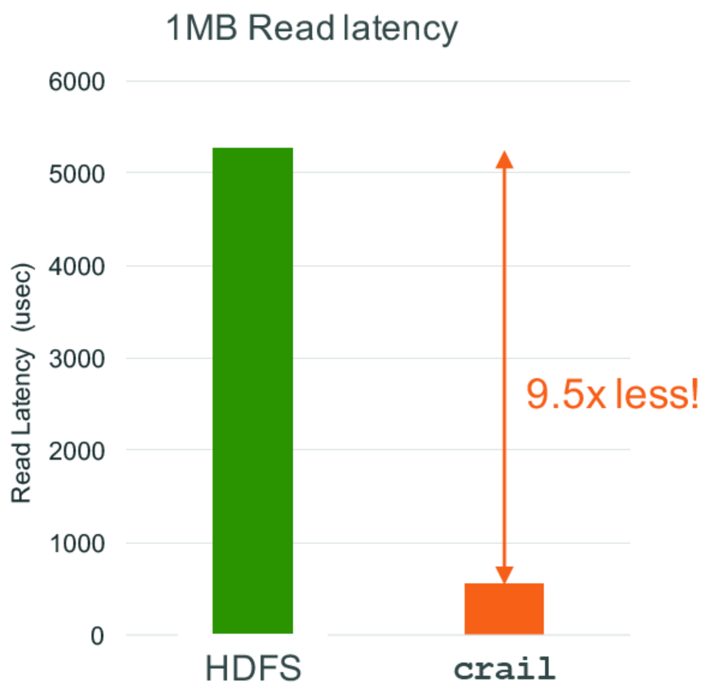
Storage Disaggregation

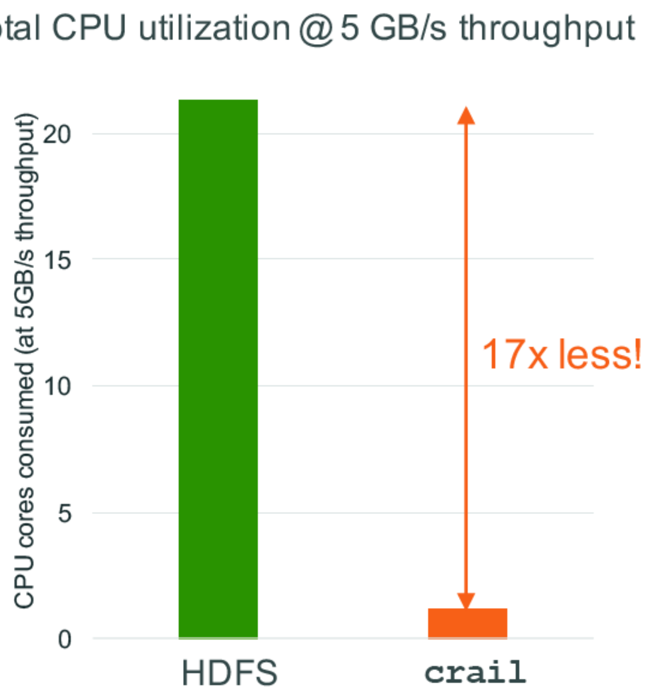


- Why disaggregation?
  - Independent scaling of compute and storage
  - Higher utilization due to less fragmentation
  - Easier maintenance
- Challenges:
  - Systems like Hadoop/Spark have been designed for local storage
  - But: new <u>fast networks</u> may permit storage disaggregation



IBM Flashsystem: Crail vs HDFS





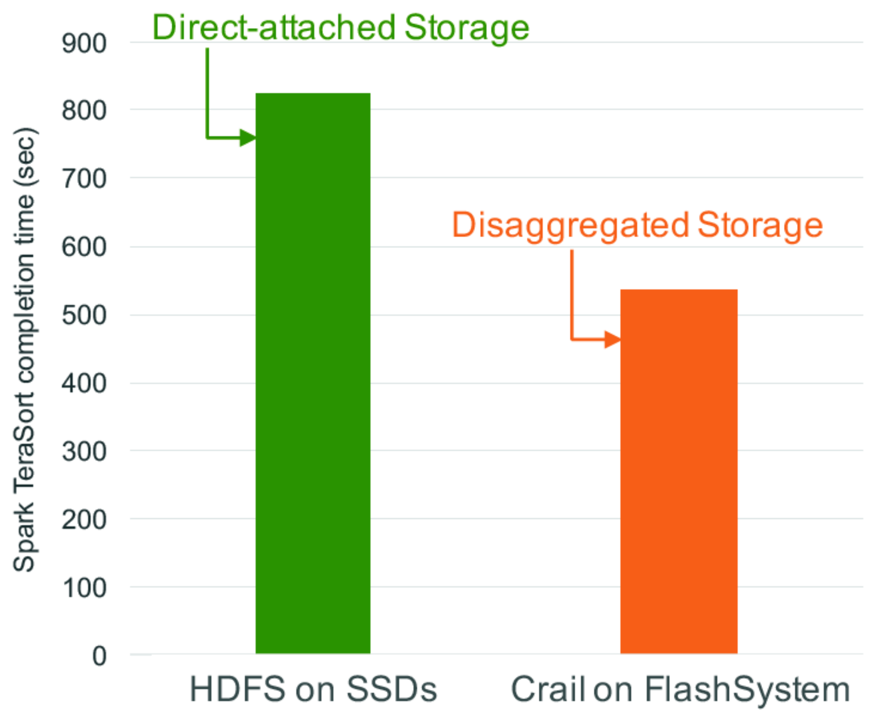
HDFS setup
- 10 node cluster
- 56 Gbit Infiniband network
- 2 x 1TB SSDs / node
- No replication

`crail` setup
- 10 node cluster
- 56 Gbit Inifiniband network
- 1 x FlashSystem 840
  - 8 Flash cards
  - 23TB usable capacity

The two systems have the same bandwidth from Flash (~10 GB/s) and about the same total capacity.

Crail + FlashSystem enables efficient, high-performance disaggregated storage for Hadoop & Spark

IBM Flashsystem: TeraSort with HDFS vs Crail



Experimental Setup

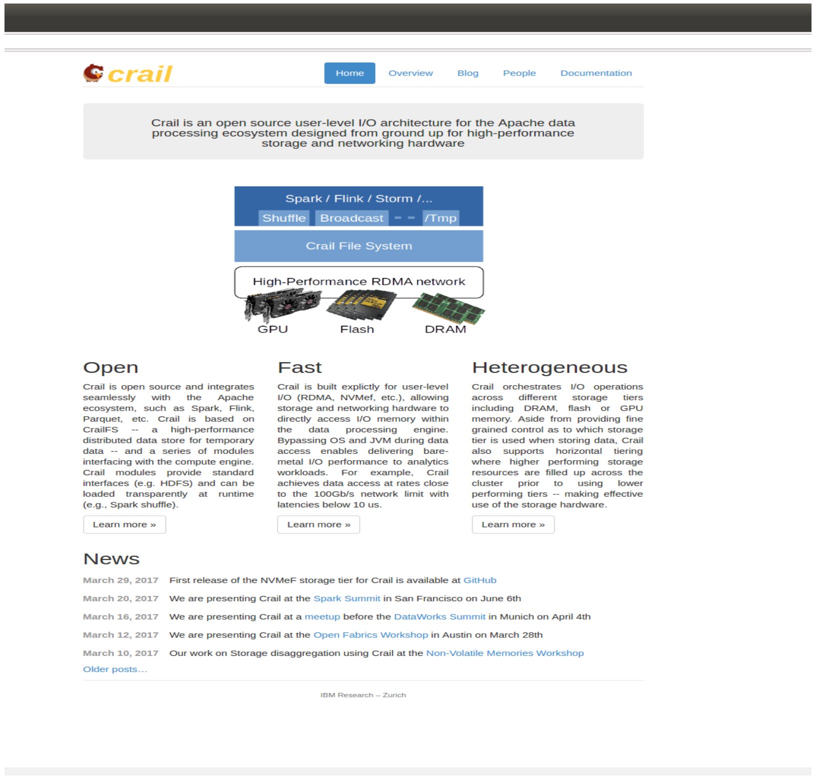
- Sorting 400GB of data using Spark
- HDFS setup
  - 10 node cluster, 56 Gbit Infiniband network
  - 2 x 1TB SSDs / node, 2-way replication
  - HDFS is using host memory (OS page cache)
- `crail` setup
  - 10 node cluster, 56 Gbit Inifiniband network
  - 1 x FlashSystem 840 (8 Flash cards, 23TB usable)
  - Crail is not using host memory
- The two systems have the same bandwidth from Flash (~10 GB/s) and about the same total Flash capacity.

Crail + FlashSystem achieves 40% performance improvement with lower TCO and all the benefits of disaggregation

Crail is Open Source!

www.crail.io

https://github.com/zrlio



Crail is an open source user-level I/O architecture for the Apache data processing ecosystem designed from ground up for high-performance storage and networking hardware

Open

Crail is open source and integrates seamlessly with the Apache ecosystem, such as Spark, Flink, Parquet, etc. Crail is based on CrailFS -- a high-performance distributed data store for temporary data -- and a series of modules interfacing with the compute engine. Crail modules provide standard interfaces (e.g. HDFS) and can be loaded transparently at runtime (e.g., Spark shuffle).

Learn more »

Fast

Crail is built explictly for user-level I/O (RDMA, NVMef, etc.), allowing storage and networking hardware to directly access I/O memory within the data processing engine. Bypassing OS and JVM during data access enables delivering bare-metal I/O performance to analytics workloads. For example, Crail achieves data access at rates close to the 100Gb/s network limit with latencies below 10 us.

Learn more »

Heterogeneous

Crail orchestrates I/O operations across different storage tiers including DRAM, flash or GPU memory. Aside from providing fine grained control as to which storage tier is used when storing data, Crail also supports horizontal tiering where higher performing storage resources are filled up across the cluster prior to using lower performing tiers -- making effective use of the storage hardware.

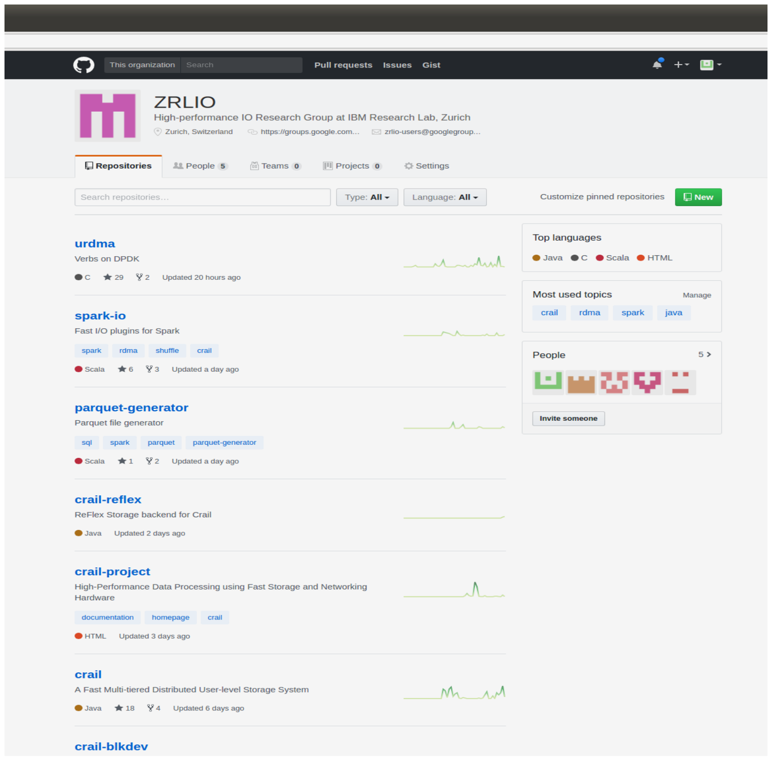
Learn more »

News

March 29, 2017	First release of the NVMeF storage tier for Crail is available at GitHub
March 20, 2017	We are presenting Crail at the Spark Summit in San Francisco on June 6th
March 16, 2017	We are presenting Crail at a meetup before the DataWorks Summit in Munich on April 4th
March 12, 2017	We are presenting Crail at the Open Fabrics Workshop in Austin on March 28th
March 10, 2017	Our work on Storage disaggregation using Crail at the Non-Volatile Memories Workshop

Older posts…

IBM Research – Zurich



ZRLIO

High-performance IO Research Group at IBM Research Lab, Zurich

Zurich, Switzerland · https://groups.google.com... · zrlio-users@googlegroup...

urdma
Verbs on DPDK
● C ★ 29 2 Updated 20 hours ago

spark-io
Fast I/O plugins for Spark
spark rdma shuffle crail
● Scala ★ 6 3 Updated a day ago

parquet-generator
Parquet file generator
sql spark parquet parquet-generator
● Scala ★ 1 2 Updated a day ago

crail-reflex
ReFlex Storage backend for Crail
● Java Updated 2 days ago

crail-project
High-Performance Data Processing using Fast Storage and Networking Hardware
documentation homepage crail
● HTML Updated 3 days ago

crail
A Fast Multi-tiered Distributed User-level Storage System
● Java ★ 18 4 Updated 6 days ago

crail-blkdev

Top languages
● Java ● C ● Scala ● HTML

Most used topics Manage
crail rdma spark java

People 5

Related Work

Three classes of related work:

- New Data Processing Systems for High-Performance Network & Storage Hardware
  - FARM, RamCloud, HERD, etc

    Fast, but require applications to be written from scratch
- Updates/patches to existing Systems
  - Ohio Spark/Hadoop Distro

    Slow because no radical changes possible: fetrofitting RDMA/Flash integration into existing file/socket based I/O stacks
- Memory/Flash caches/stores
  - Example: Tacyon

    Slow because not designed for high-performance hardware

Conclusion

Today's open source analytics stacks:

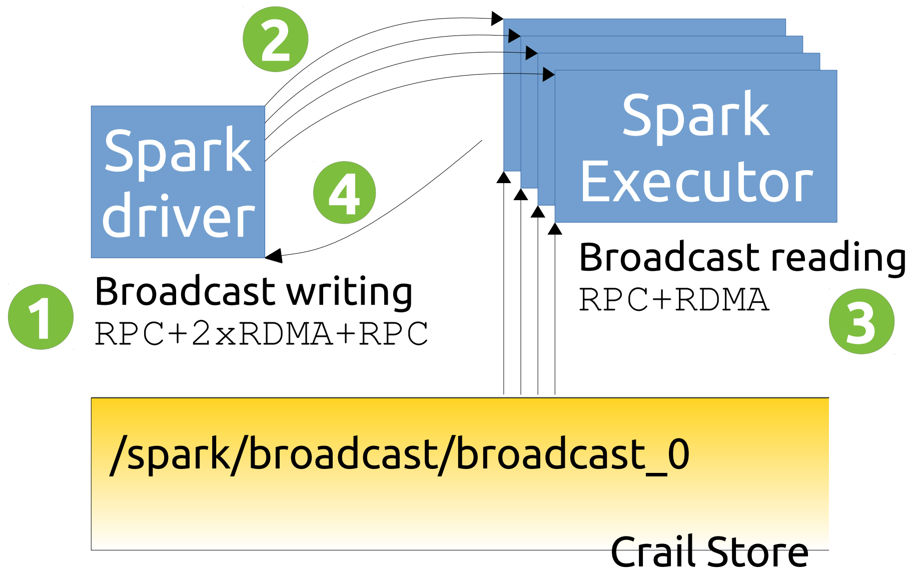
- Existing analytics stacks designed for yesterday's commodity hardware
- Performance on high-end hardware inhibited by heavy-layered stack architecture

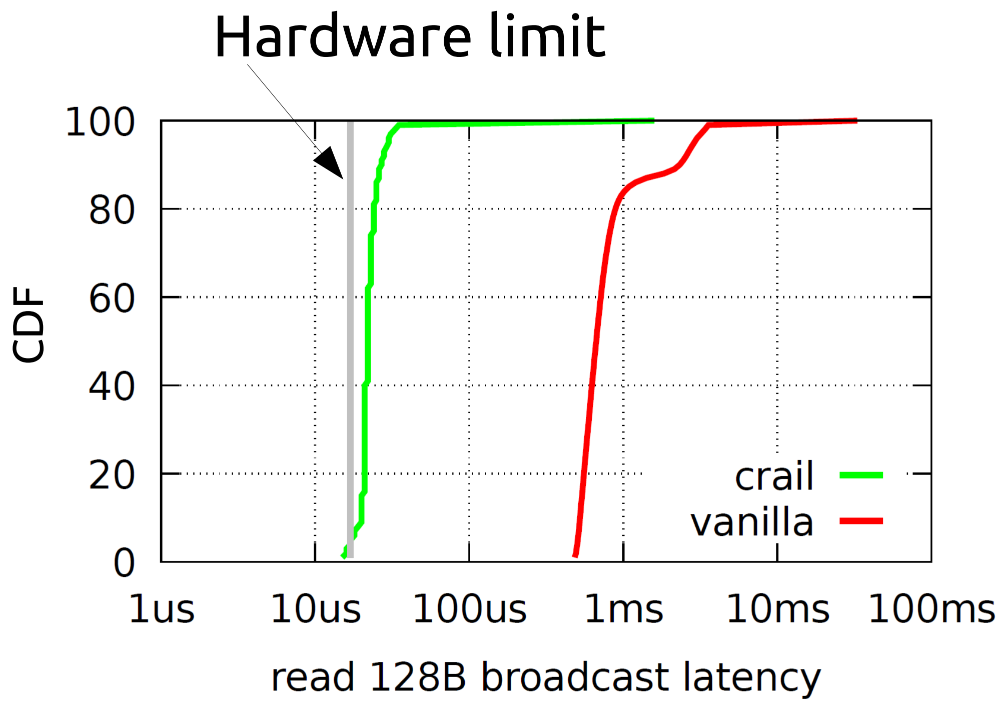
The Crail Approach:

- Radical re-design of I/O (network & storage) for analytics by exploiting modern hardware
  - RDMA, NVMe & NVMe over fabrics
- Enable high-performance disaggregated storage for analytics
- Extend Spark operation to take advantage of Crail
- Crail is open source: www.crail.io

Backup

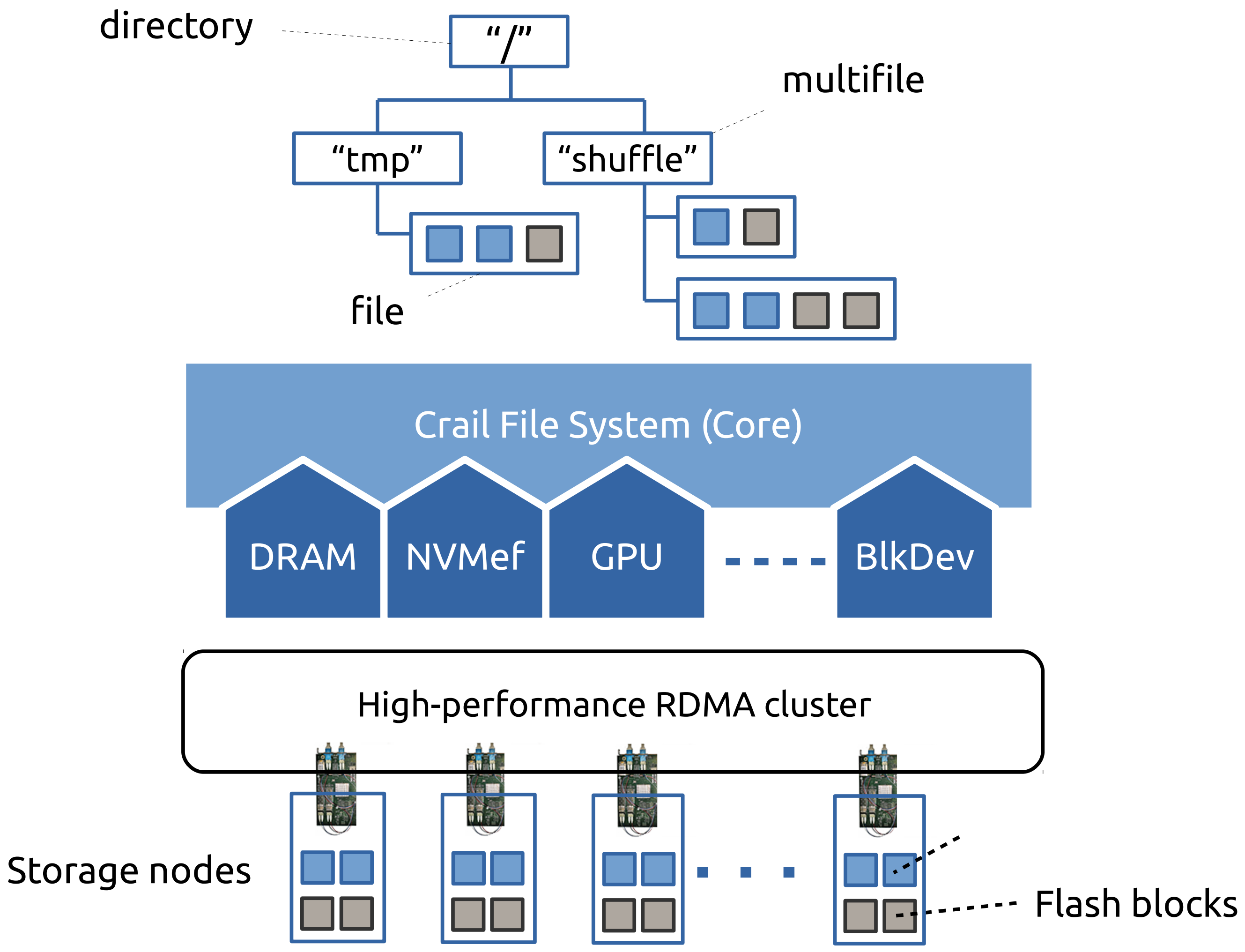
Spark/Crail Broadcast



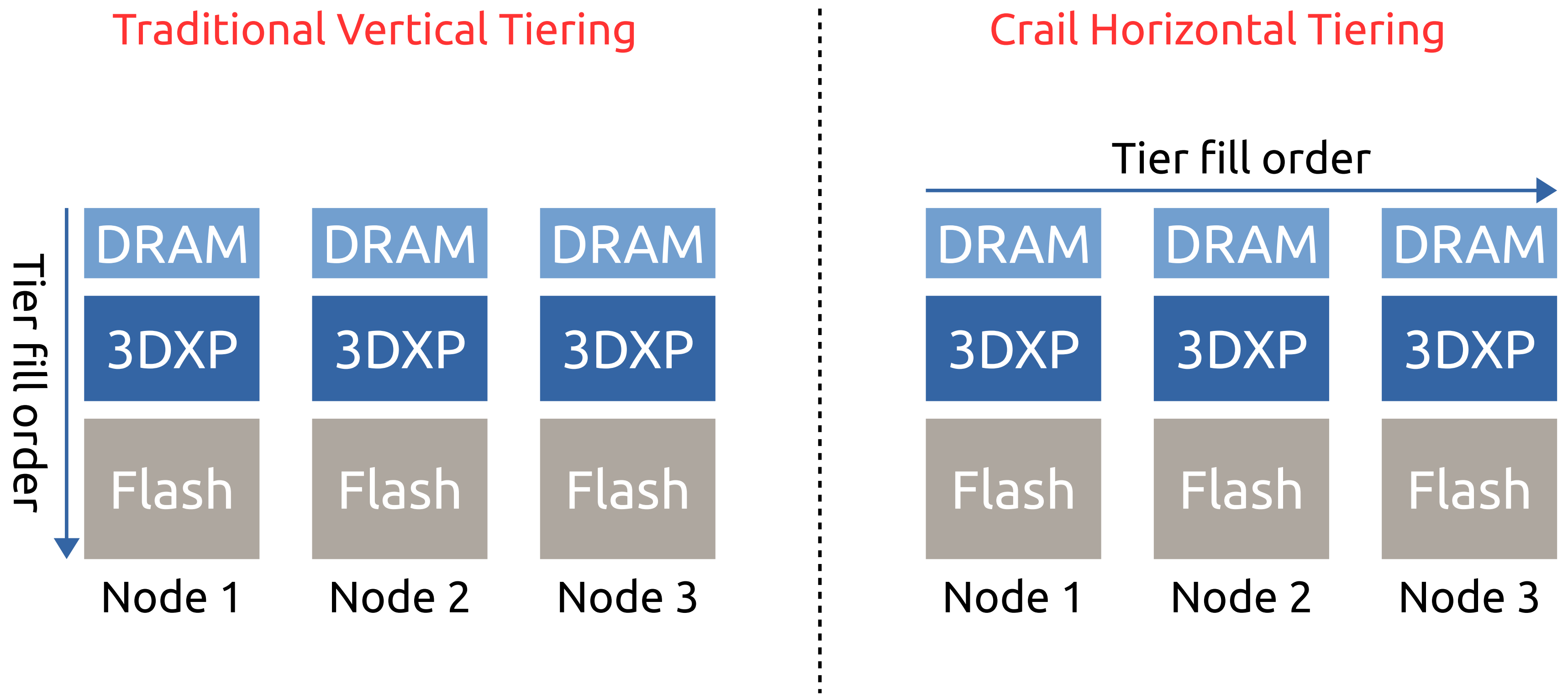


```
val bcVar = sparkContext.Broadcast(new Array[Byte](128))
sparkContext.parallelize(1 to tasks, tasks).map(_ => {
  bcVar.value.length
}).count
```

The Crail Store



Crail Storage Tiering



With horizontal tiering, higher-performing tiers are filled up across the cluster prior to using lower performing tiers