Data Processing at the Speed of 100 Gbps@Apache Crail (Incubating)

http://crail.incubator.apache.org/

Animesh Trivedi
IBM Research, Zurich

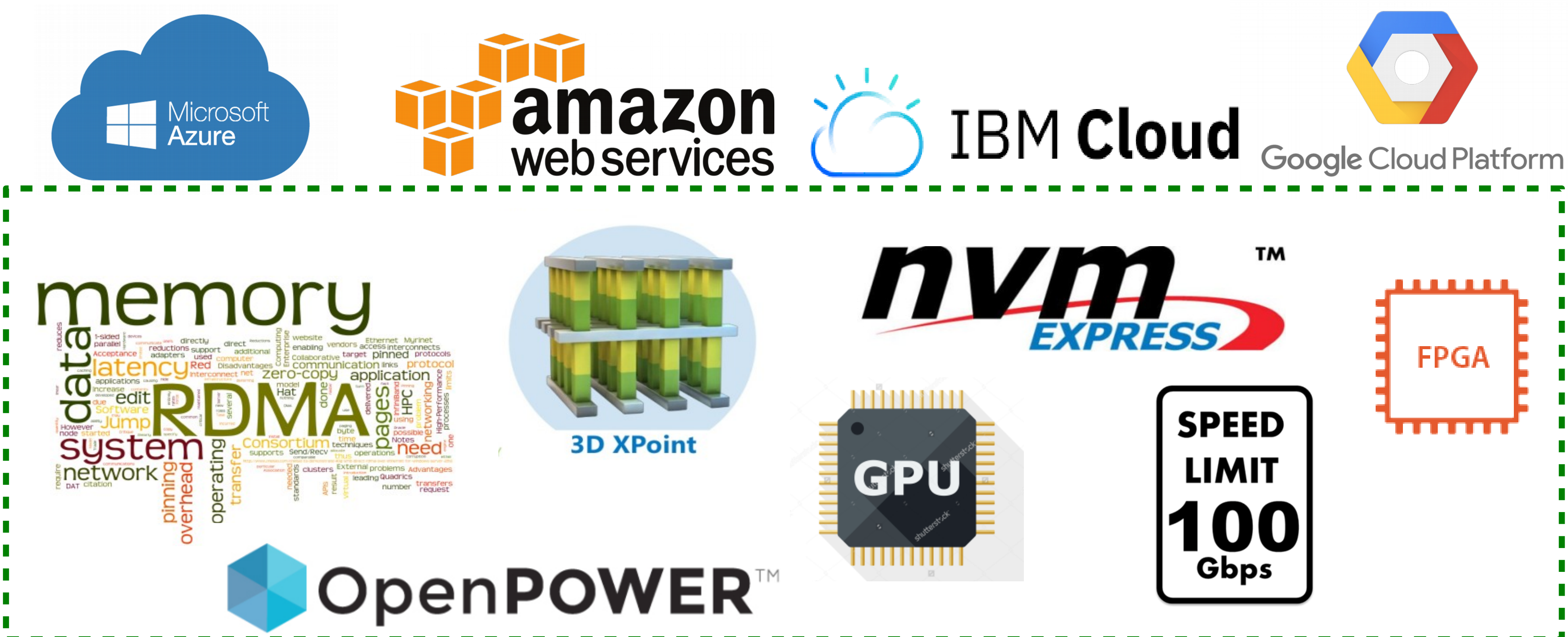
The Performance Landscape

Trend 1: The I/O performance is increasing dramatically

	2010	2018	2018-
Storage	100 MB/s (HDDs)	1,000 MB/s (SSDs, NVMe)	10,000 MB/sec (3DXP)
Network	1 Gbps	10/25/40 Gbps	100/200 Gbps
CPU	1 x ~3 GHz	n x ~3 GHz	n x ~3 GHz

The Performance Landscape

Trend 2: The I/O diversity is increasing dramatically



The Key Challenge

Given the current hardware performance and diversity landscape, how can we orchestrate data movement at the speed of hardware

in other words:

" Can we feed modern data processing stacks at 100+ Gbps data speeds with 1-10 usec access latencies"

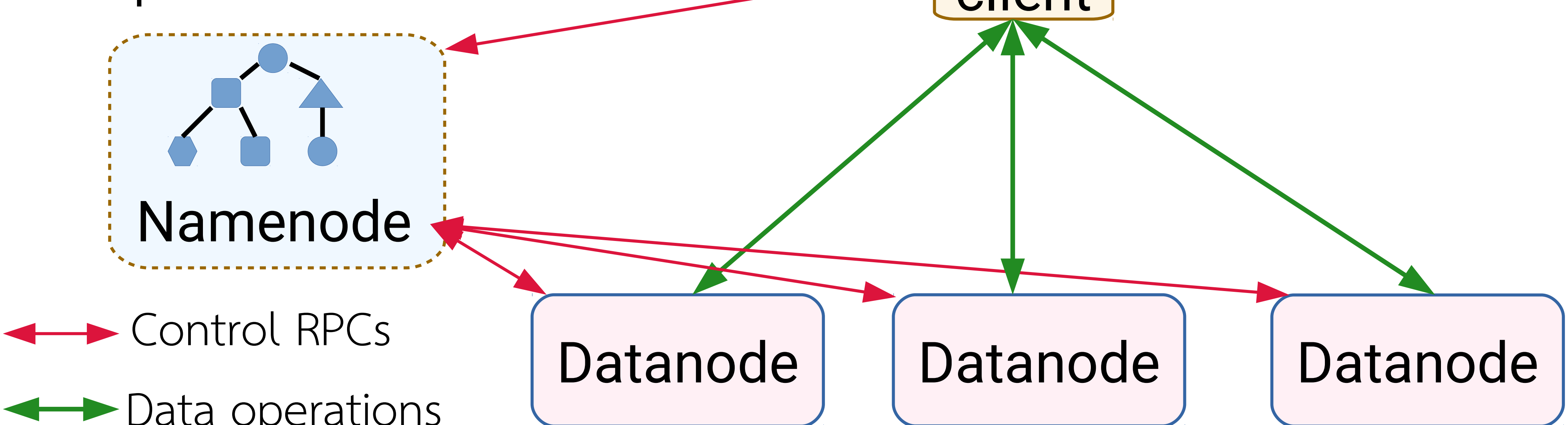
Apache Crail (Incubating)

- A distributed data "store" platform designed from scratch to "share" data at the speed of hardware
  - "store": in DRAM/Flash/3DXP, with multiple front-end APIs to storage
  - "share": intermediate data from intra-job (shuffle, broadcast) & inter-job
- Targets to speed-up workloads by accelerating data sharing
- An effort to unify isolated and incompatible efforts to integrate high-performance device in big data frameworks
  - Written in Java8 with multiple client-language support
- Started at IBM Research Lab, Zurich
- Apache Incubator project since November, 2017

System Overview – Crail Store

- access metadata from the namenode
- access data from datanodes

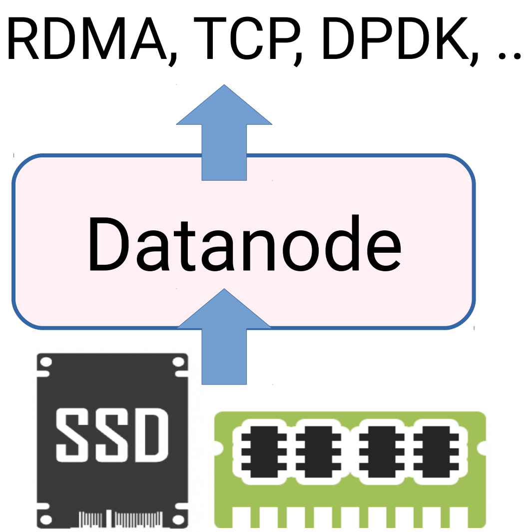
- logically centralized
- keeps track of metadata



- donate storage and network resources to Crail
- mostly dumb/stateless servers

Apache Crail: Datanode

- DataNode is responsible for exposing (storage + networking) resources to the system
  - DRAM + RDMA [1], TCP/Sockets
  - Flash + NVMeF [2]
  - Local DRAM + memcpy
  - Local flash + SPDK
- Accepts client connections and serves data
- Periodic heartbeat pings with the namenode



Apache Crail: Namenode

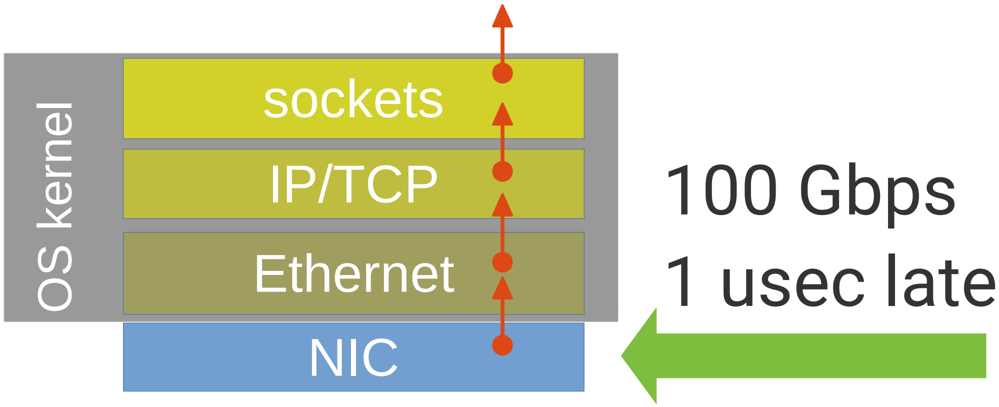
- Responsible for keeping track of storage resources in the cluster
  - Done by managing them in blocks (e.g., 1MB block size)
- Maintains a hierarchical node tree
  - directory, data files, stream files, multifiles, tables, KV
- Clients connect to the namenode to create new nodes, lookup, read, and write to nodes
  - Allocation policy (different media and node types)
  - A node can contains some blocks from DRAM and flash

Apache Crail: Clients

- Clients read and write data
  - Standalone or a part of a compute framework
  - Single writer, without holes files
  - Distributed clients often implicitly index and synchronize on the file/directory name
- Multiple-client side storage abstractions/interfaces
  - Simple hierarchical file system
  - Key-Value (KV) store
  - HDFS
  - Streaming (WiP)

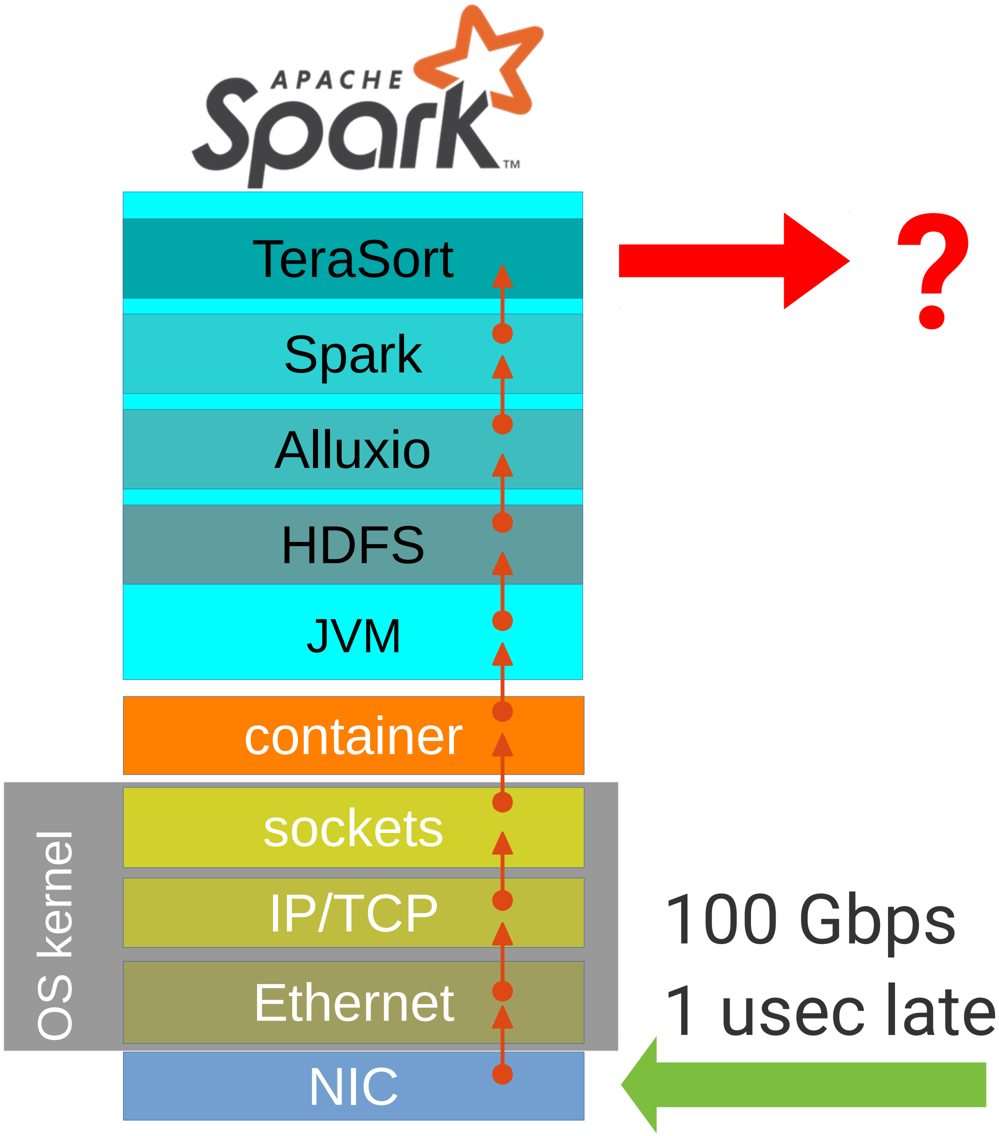
So where does the performance come from?

Performance Principles – I



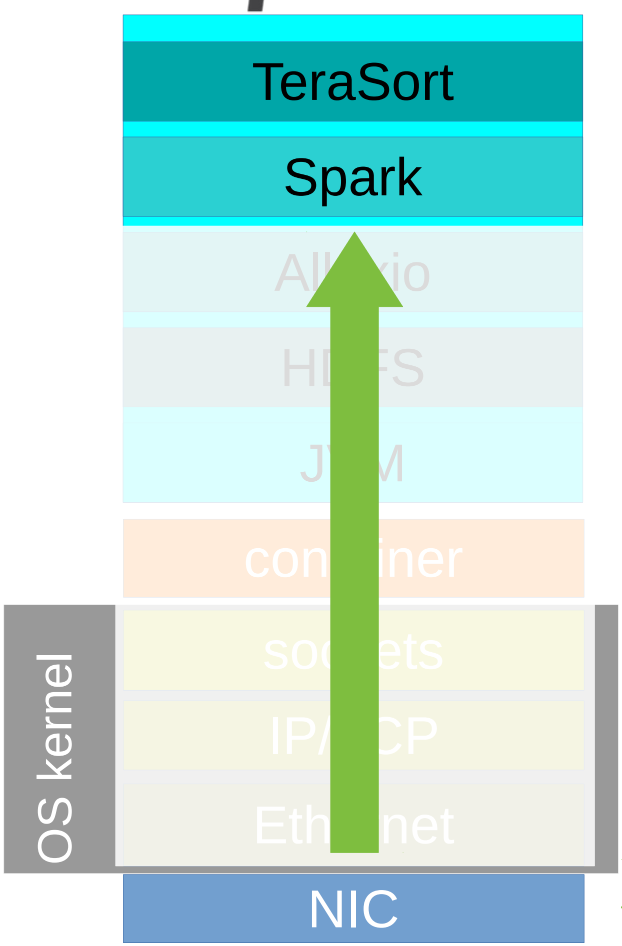
100 Gbps
1 usec latency

Performance Principles – I



100 Gbps
1 usec latency

Performance Principles – I

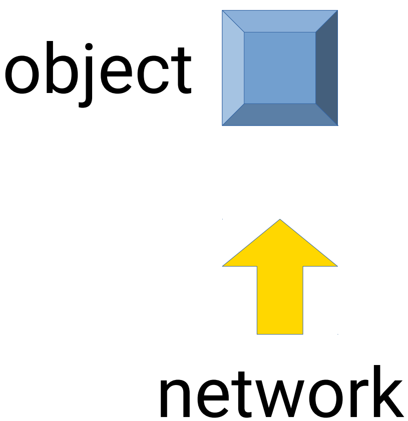


1. Use high-performance user-level I/O

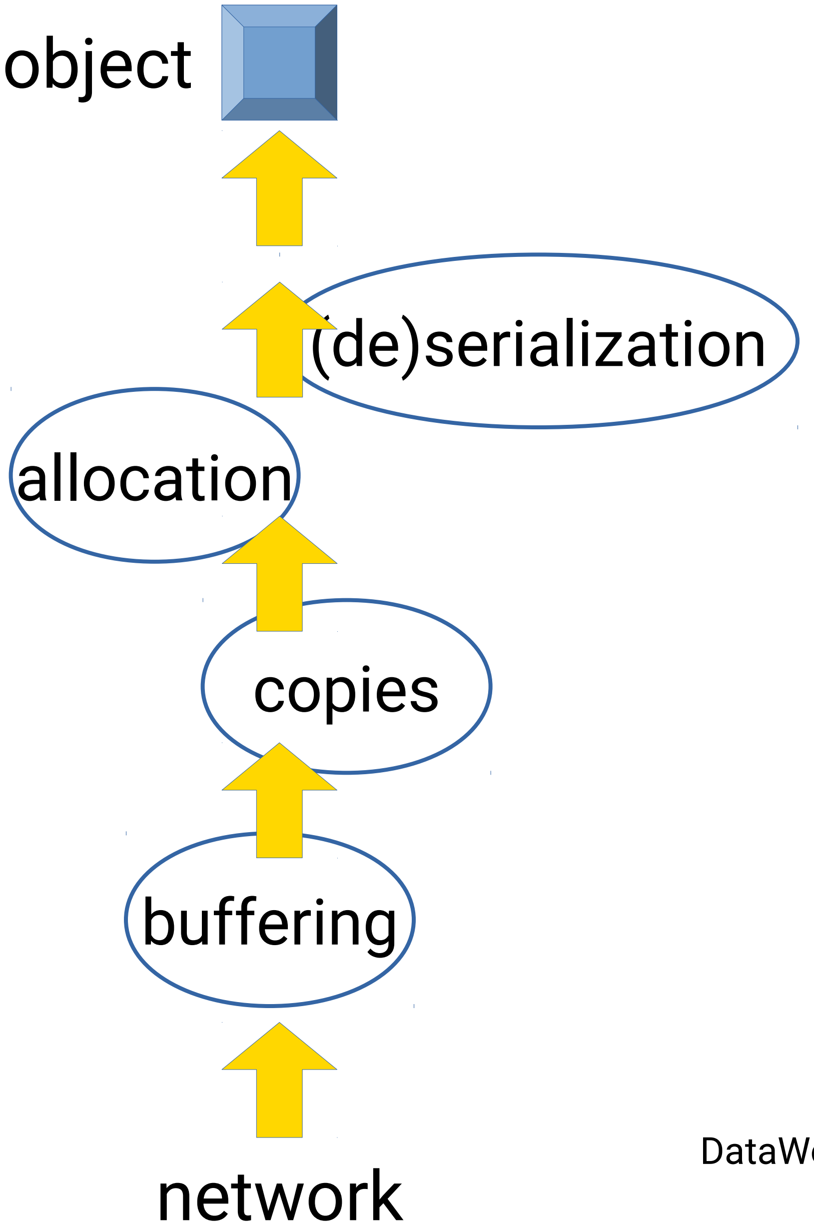
RDMA[1], NVMeF[2], SPDK/DPDK in Java
- one-sided RDMA read/write operations

100 Gbps
1 usec latency

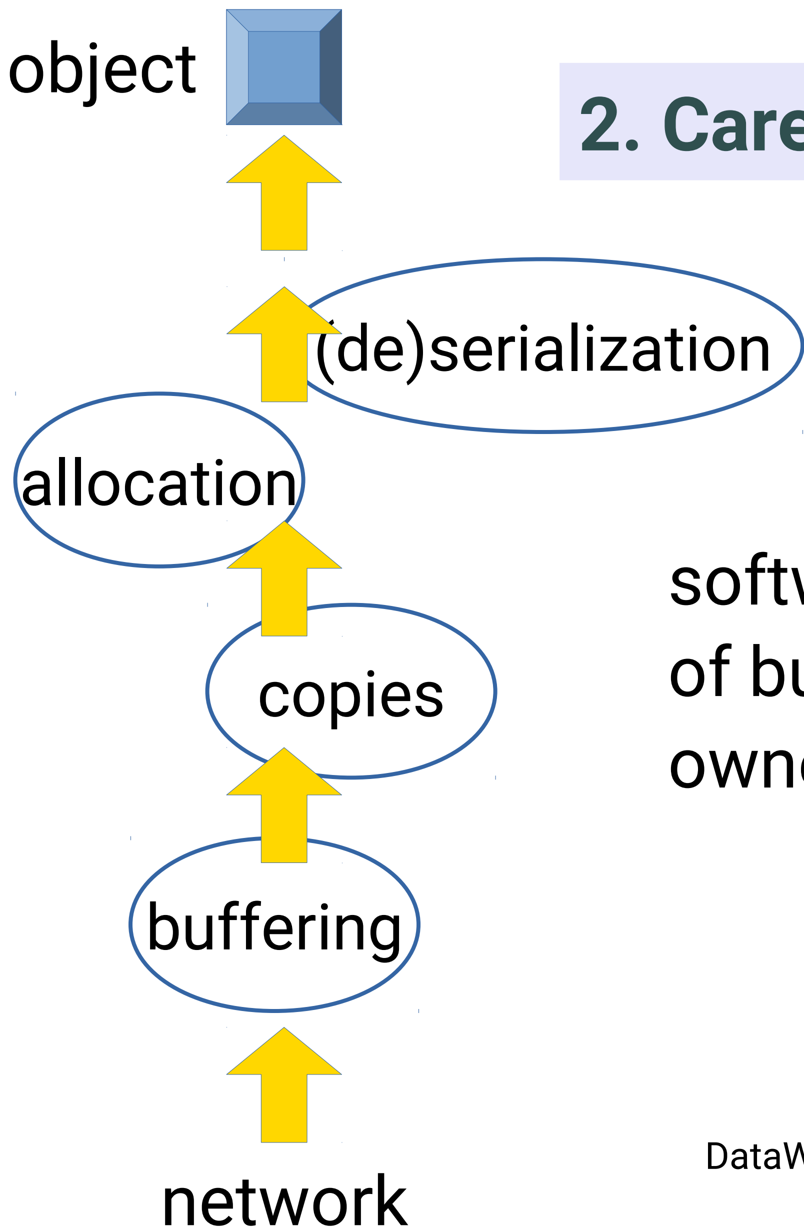
Performance Principles – II



Performance Principles – II



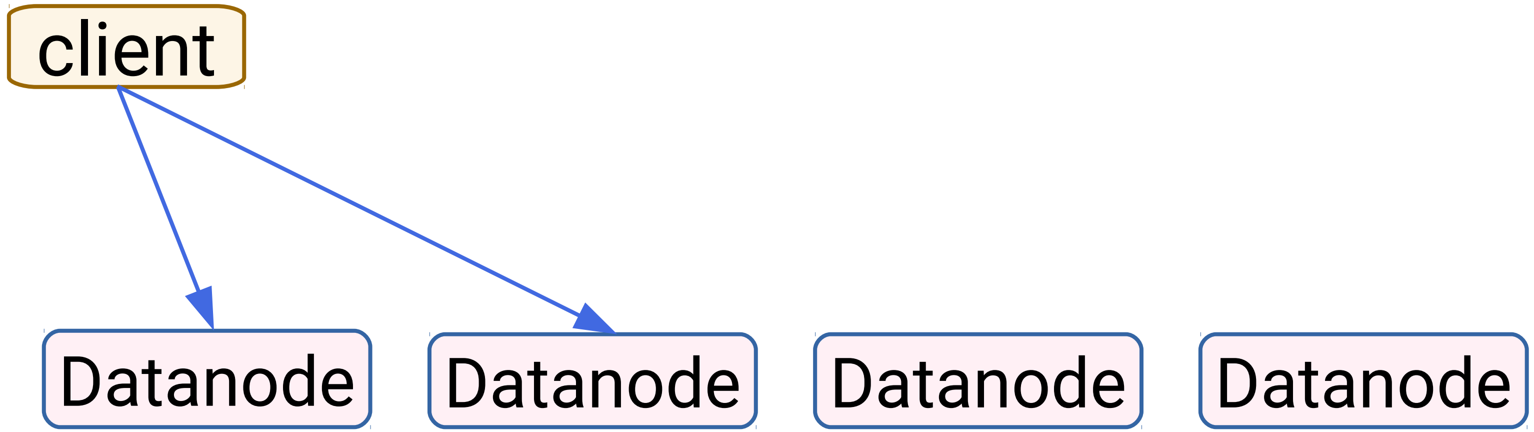
Performance Principles – II



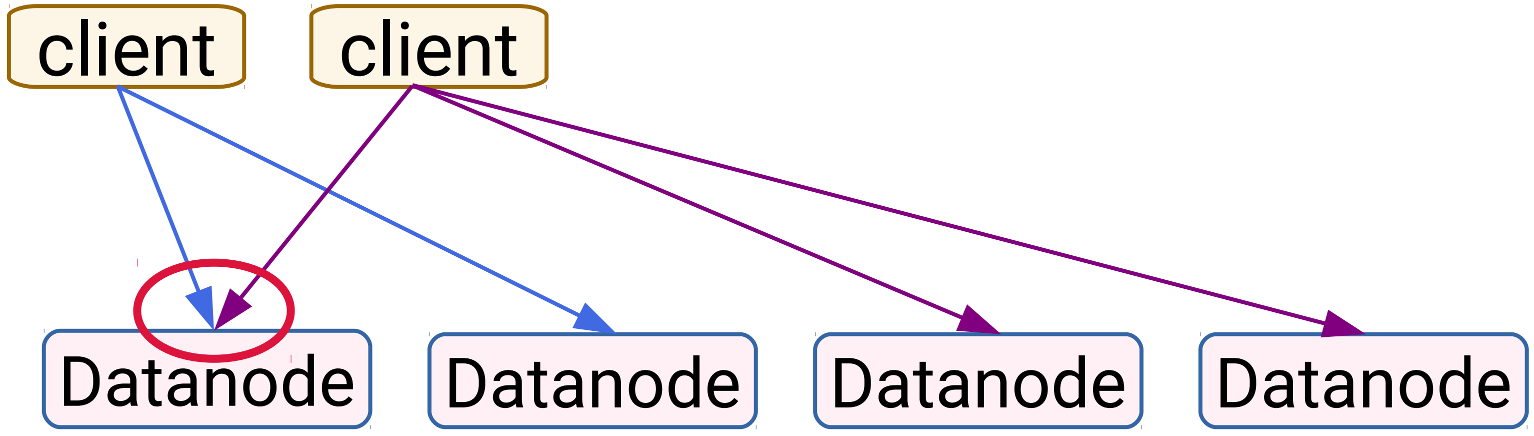
2. Careful software design for the μsec-Era

software time budgeting, pre-allocation of buffers, buffer caching, and reusing, ownership, etc.

Performance Principles – III



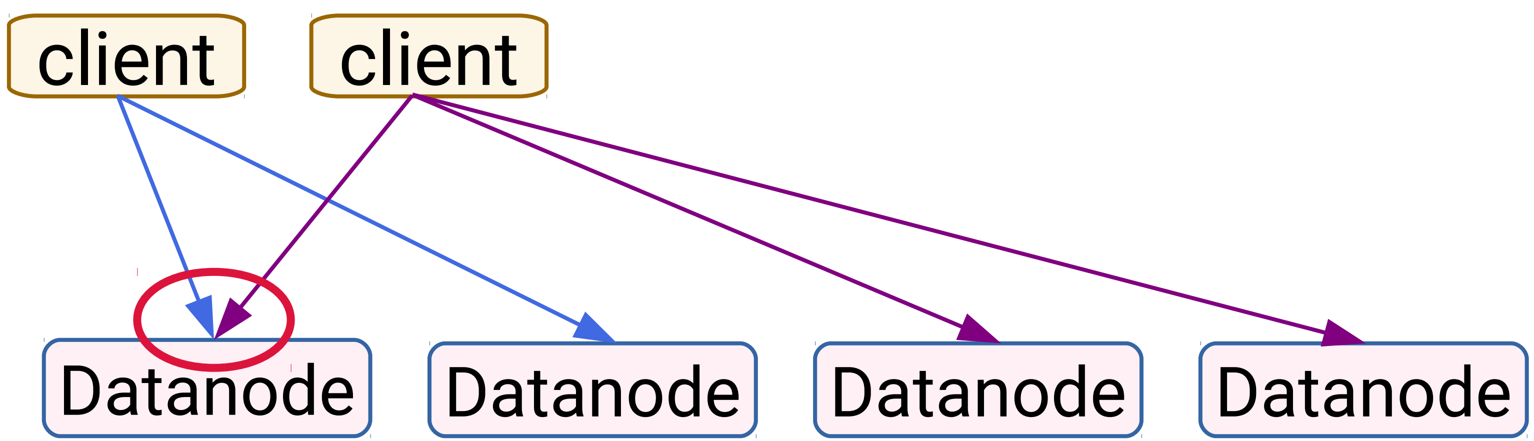
Performance Principles – III



Performance Principles – III

3. Careful data orchestration in the cluster

randomization, async requests, compute-I/O overlap, smart buffering, data allocation policies, etc.



Performance Principle - Recap

1. Use high-performance user-level I/O

2. Careful software design for the μsec-Era

3. Careful data orchestration in the cluster

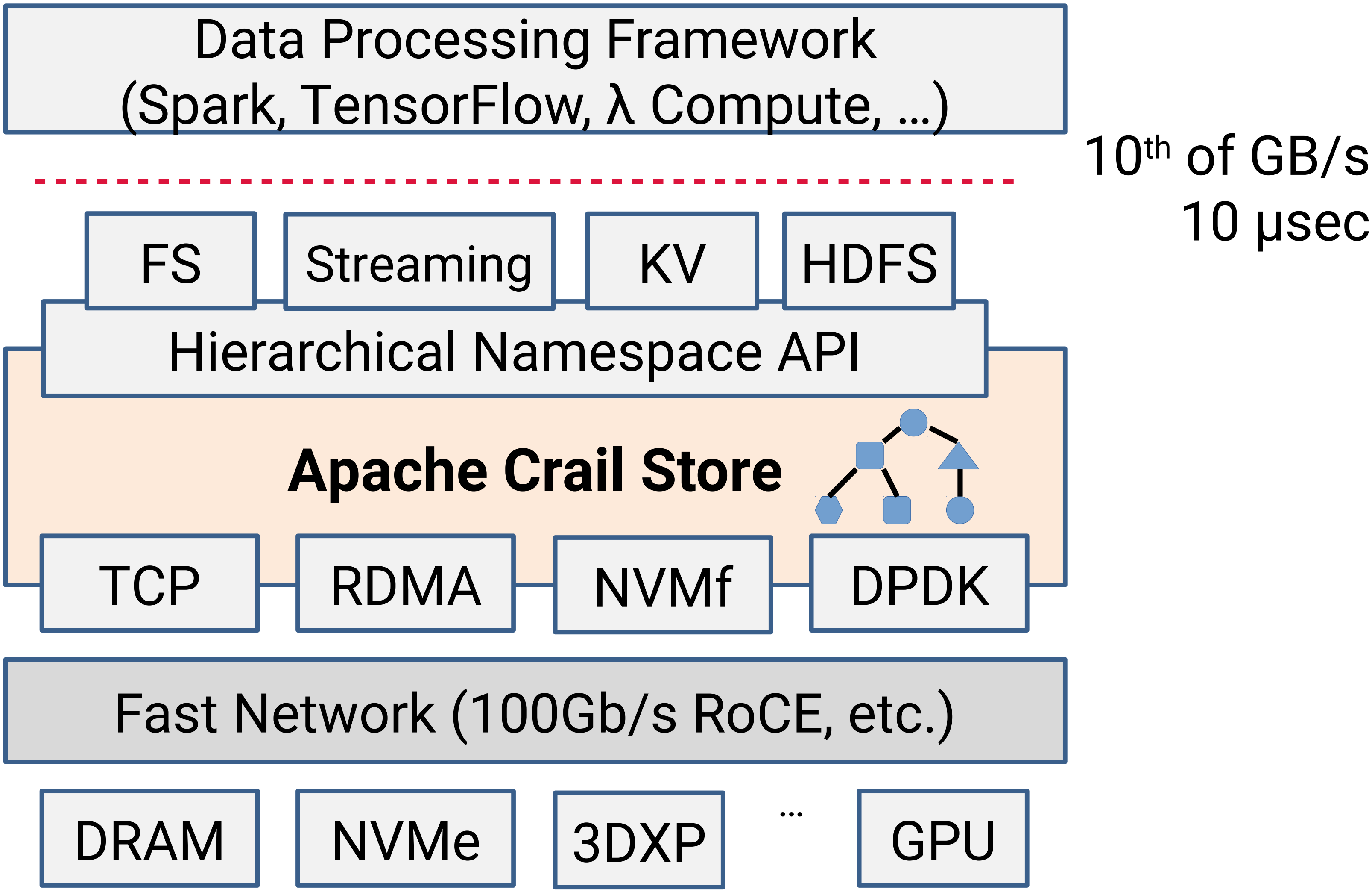
But what If I don’t have … RDMA

1. Use high-performance user-level I/O ✗

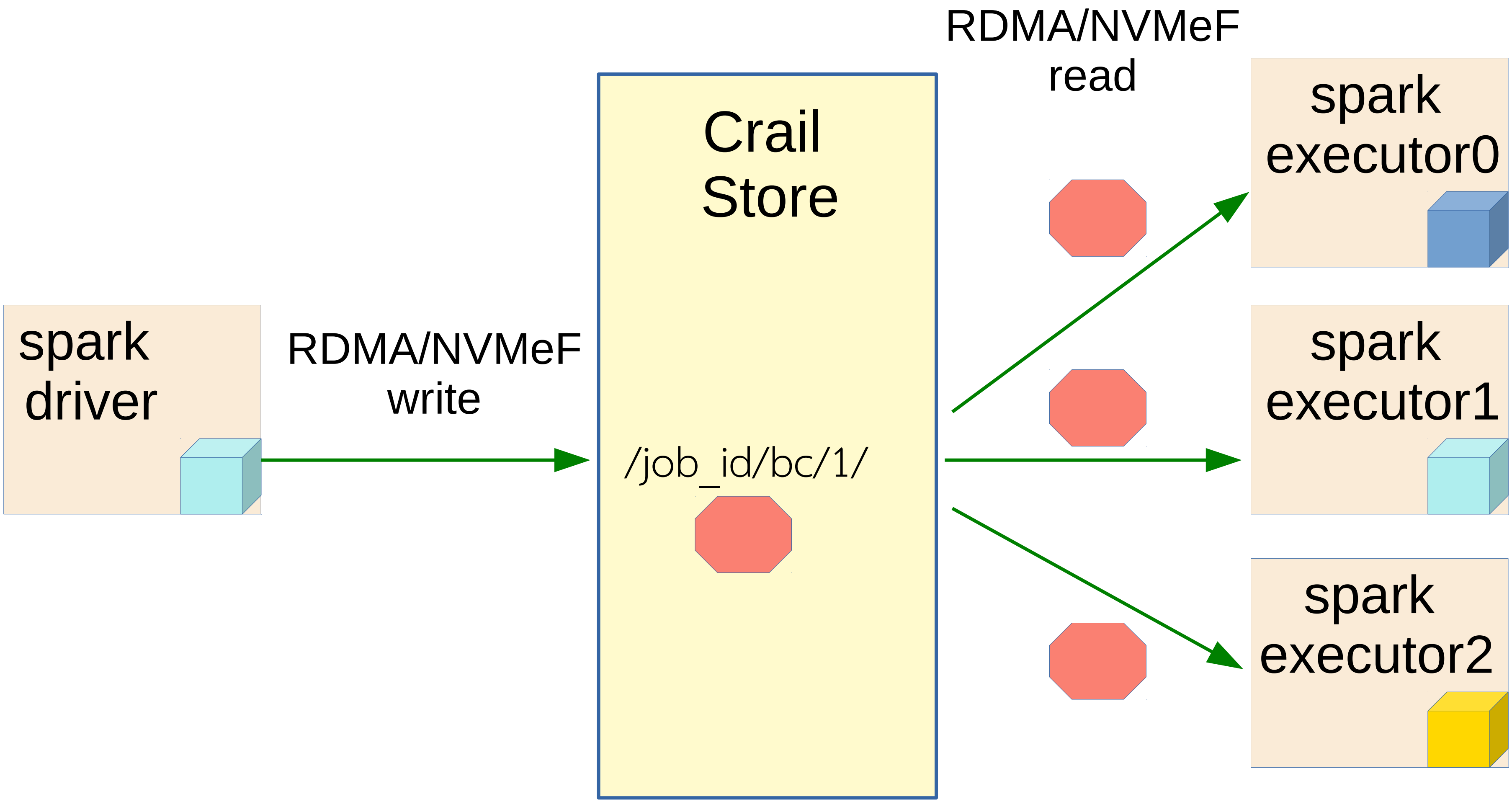
2. Careful software design for the µsec-Era ✓

3. Careful data orchestration in the cluster ✓

The Full Apache Crail (Incubating) Stack

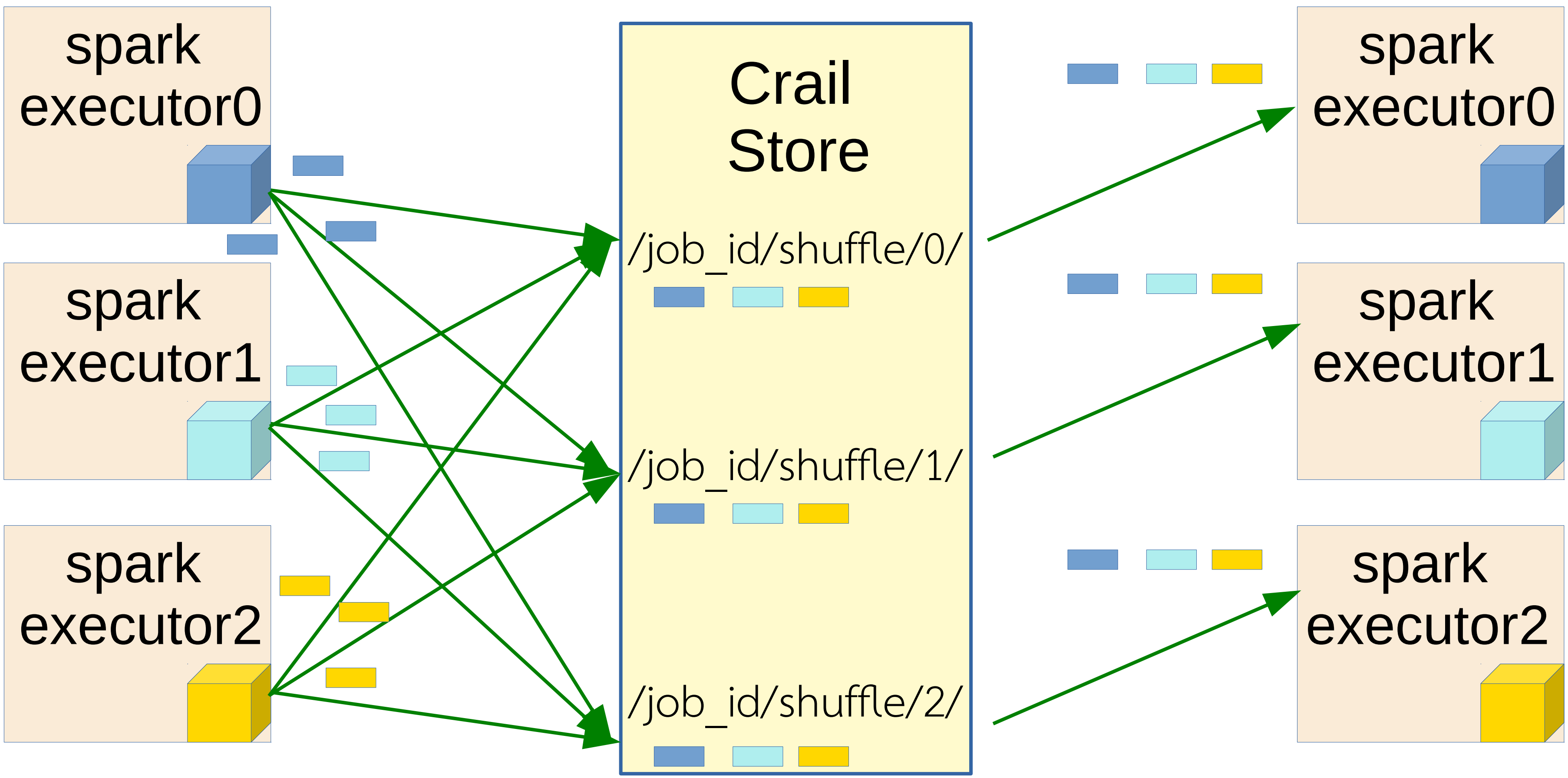


Use in Apache Spark – Broadcast



Use in Apache Spark - Shuffle

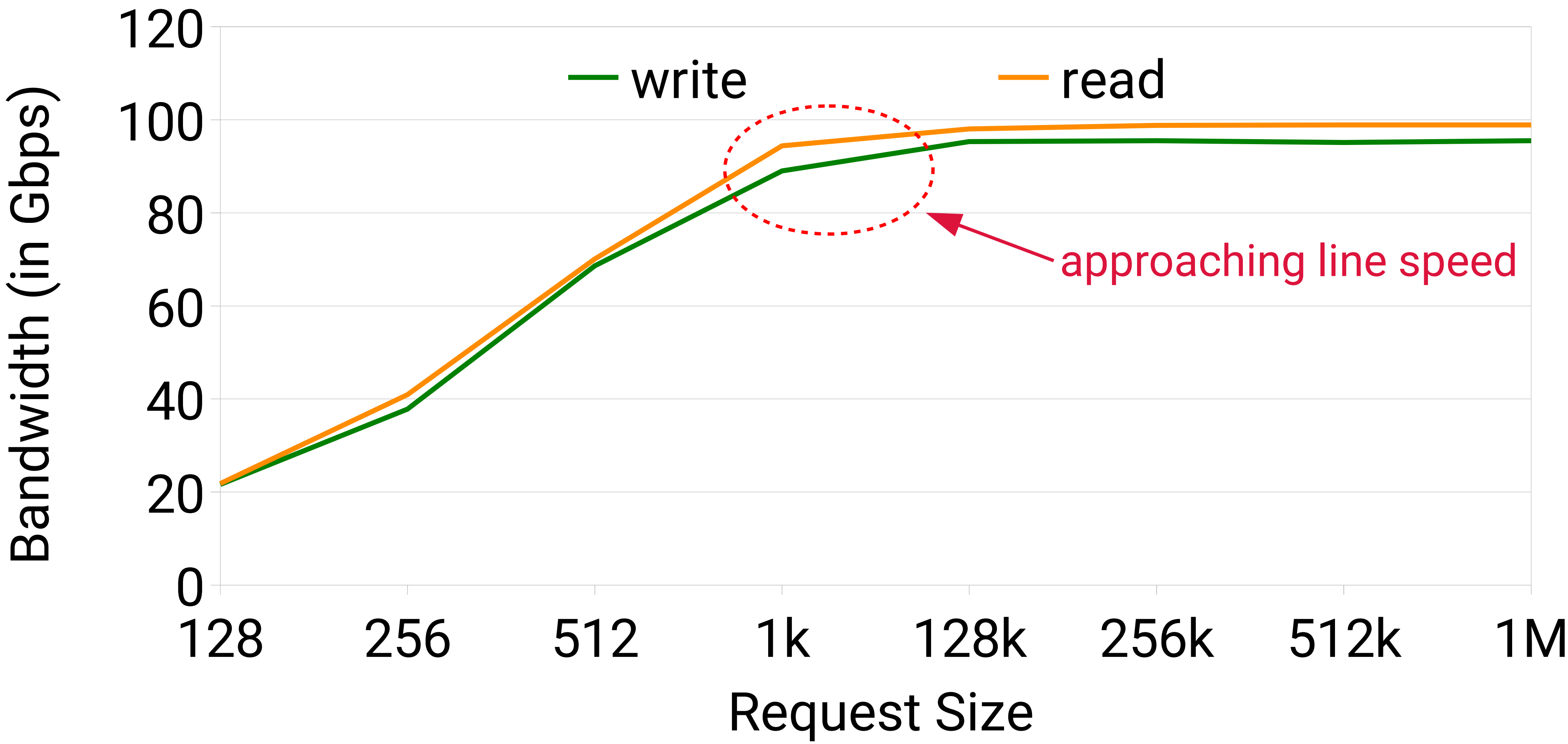
Spark Shuffle plugin [4] that write and reads data in Crail



Performance Numbers

- Baseline performance for Apache Crail Store [6]:
  - Latency, bandwidth, and IOPS numbers in *a distributed setting in the JVM*
- Crail + Spark integration, available at [4]
  - Micro-operations: Broadcast and GroupBy
  - Workloads: TeraSort and SQL JOIN
  - Mixed DRAM and Flash setting
- A mix of x86 and POWER8 machines, 100 GbE network, 256GB DRAM DDR3, and NVMe flash

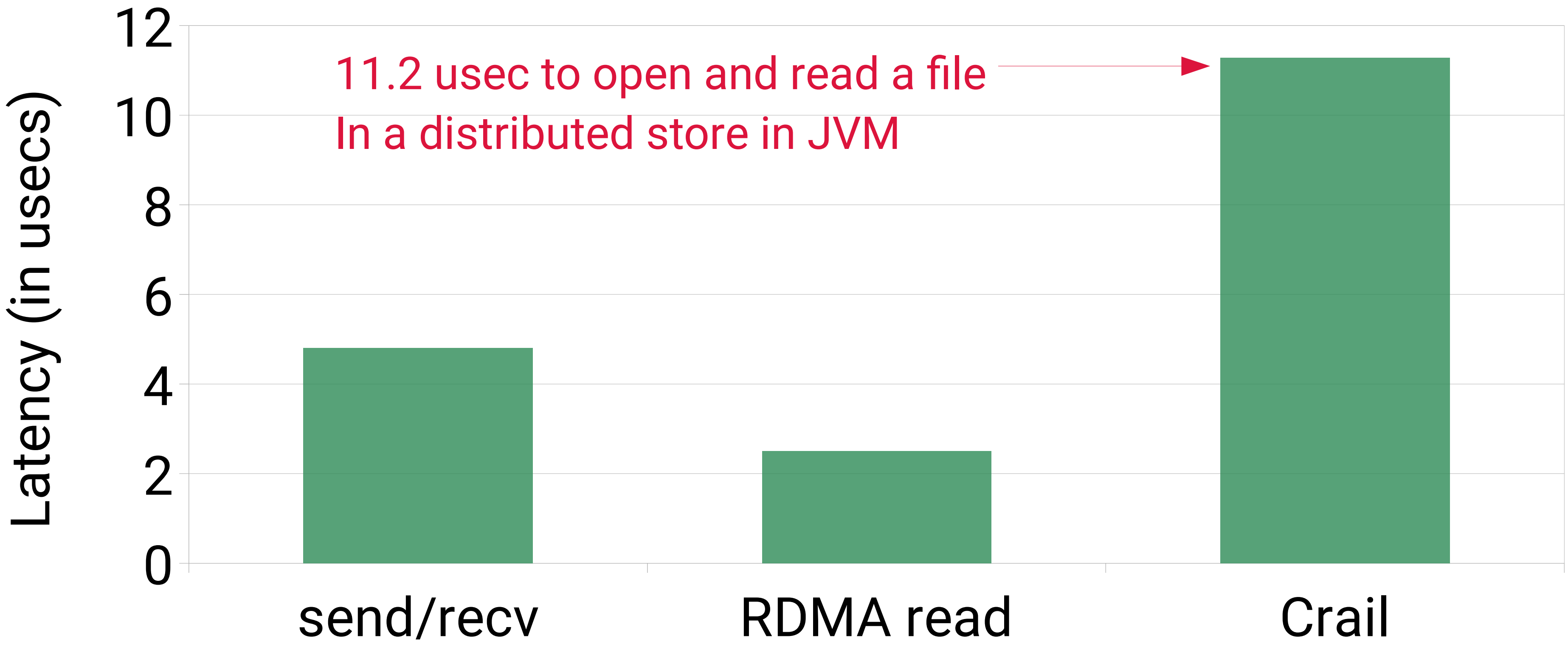
Crail Store – DRAM
”File Read Bandwidth”



Crail delivers full hardware bandwidth from DRAM

Crail Store – DRAM

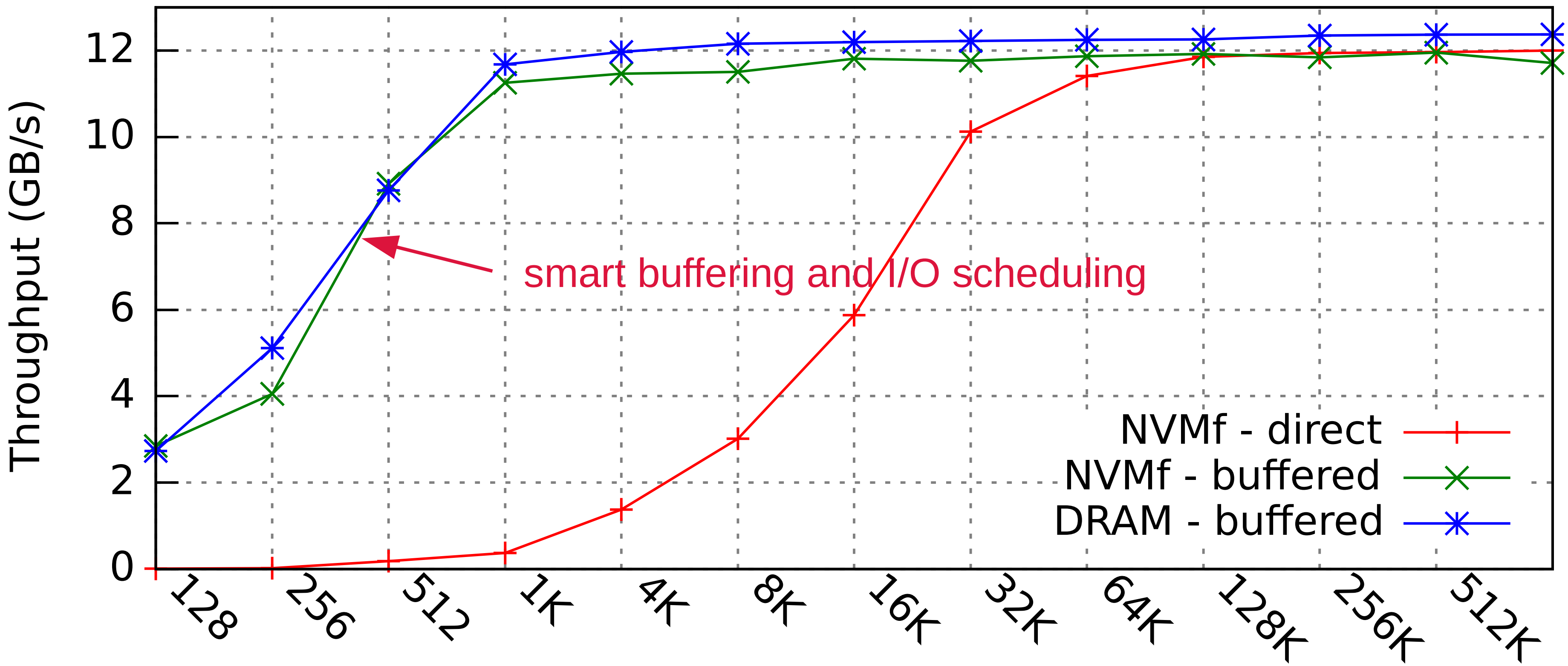
”File Read Latency”



Crail delivers ultra-low file/data access latencies in JVM

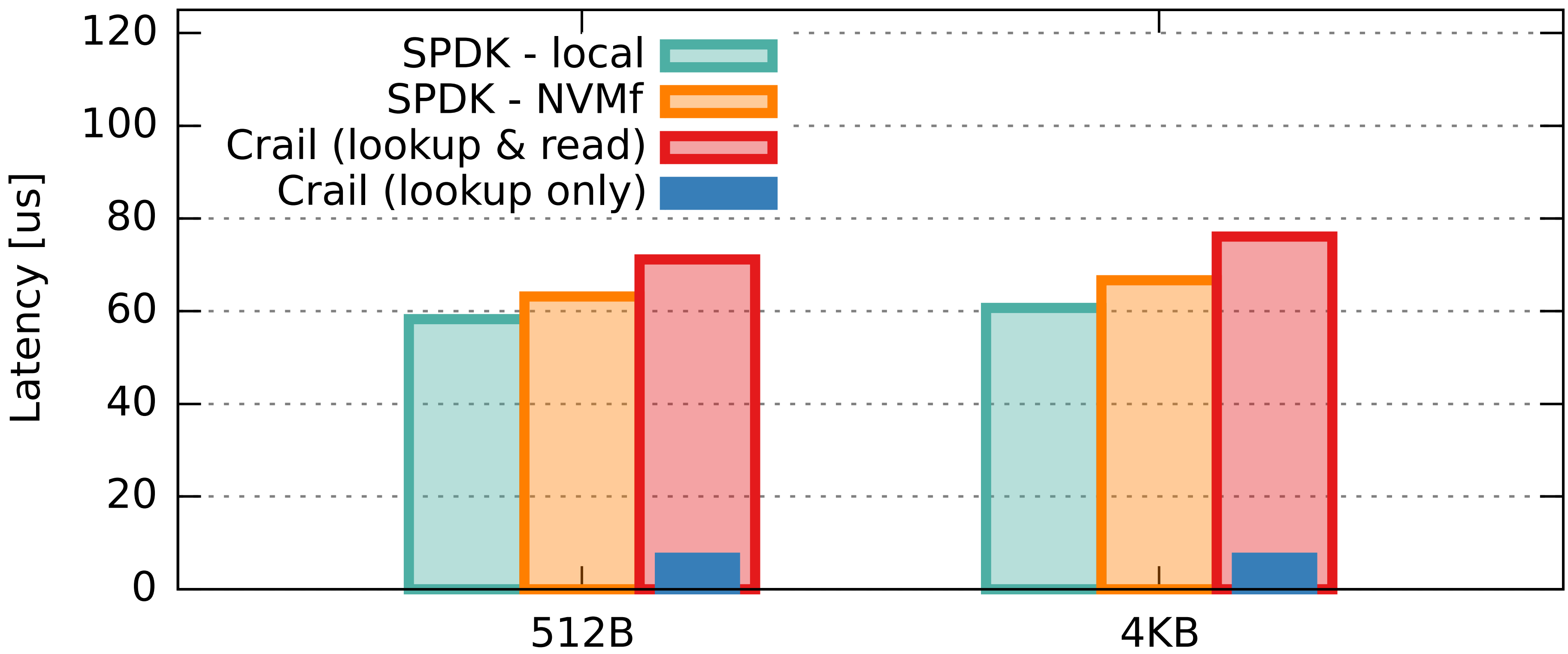
Crail Store – NVMeF

”File Read Bandwidth”



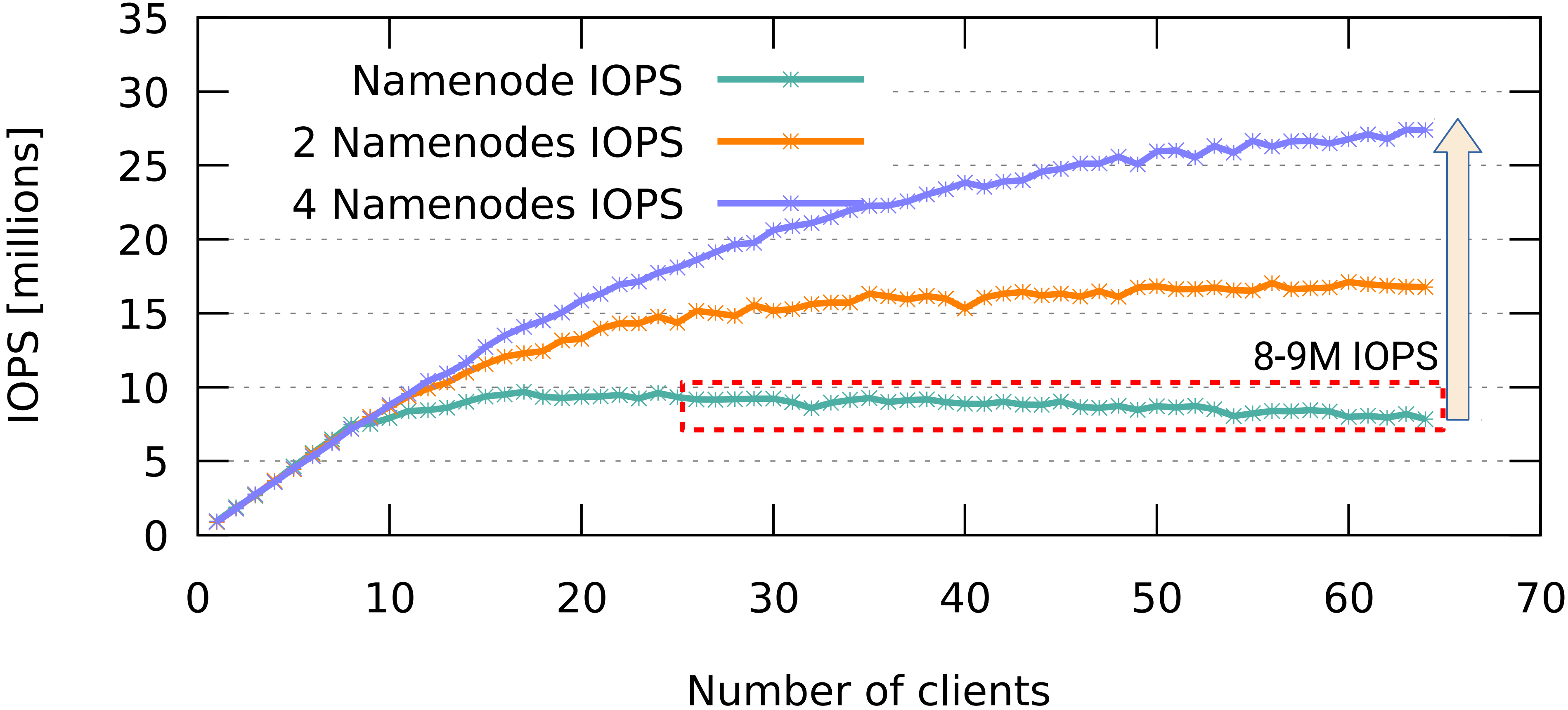
Crail delivers full hardware bandwidth from remote NVMe flash

Crail Store – NVMeF
”File Read Latency”



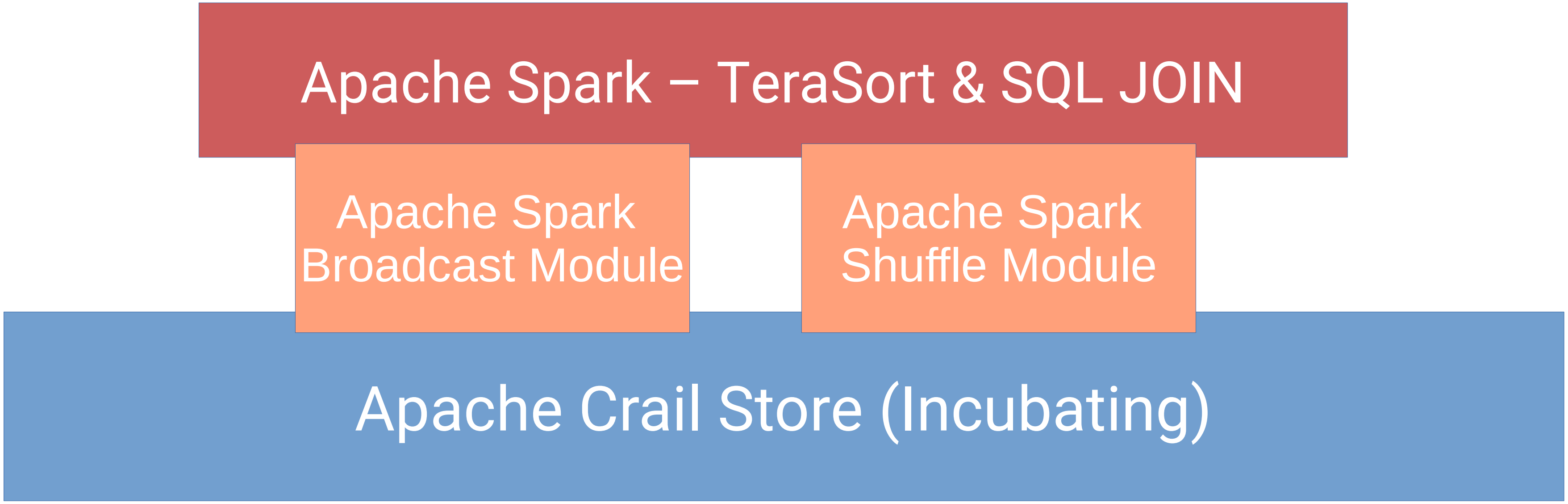
Crail delivers native NVMe access performance in JVM

Crail Store – Metadata IOPS
"getFile operation"

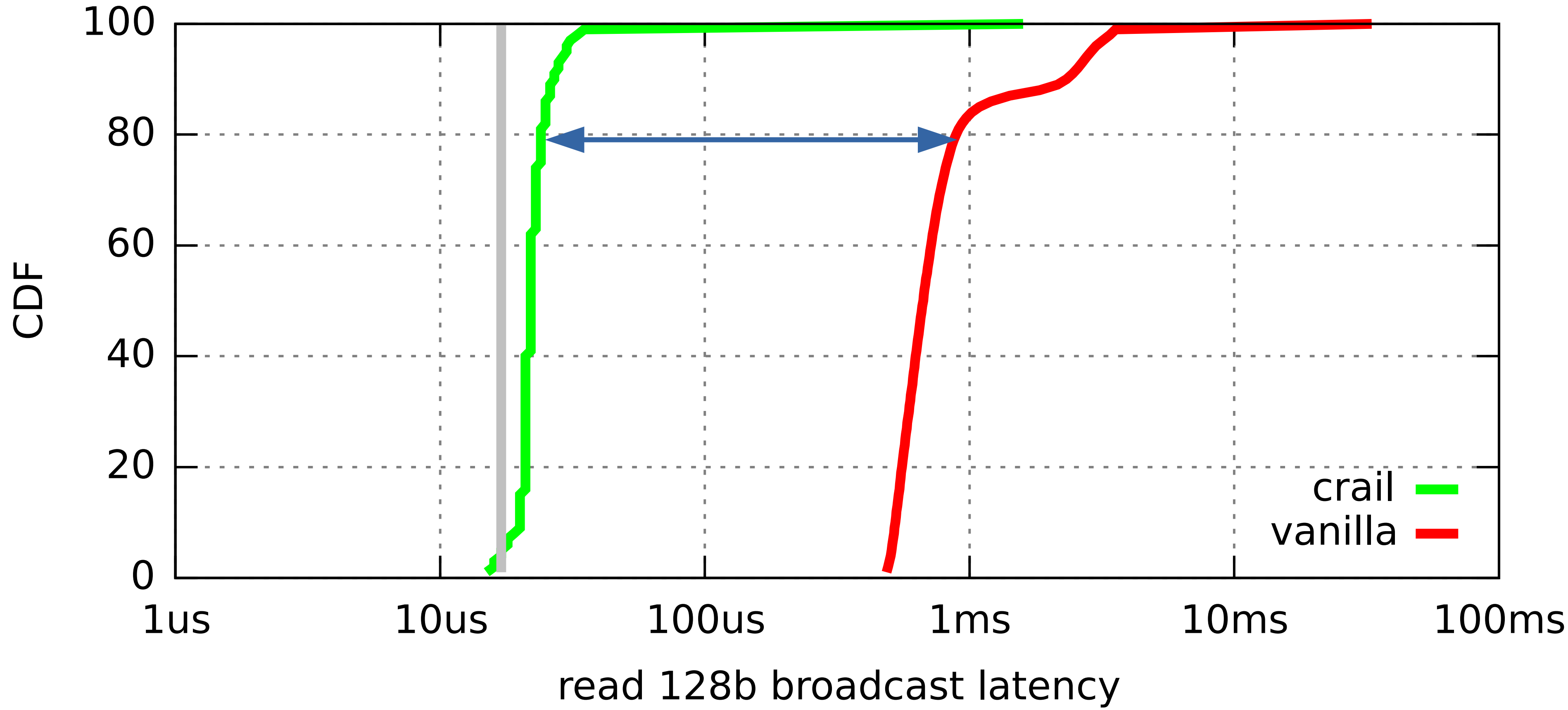


Crail offers (almost) linear scalability with Namenode ops

How does data processing-level performance look like?

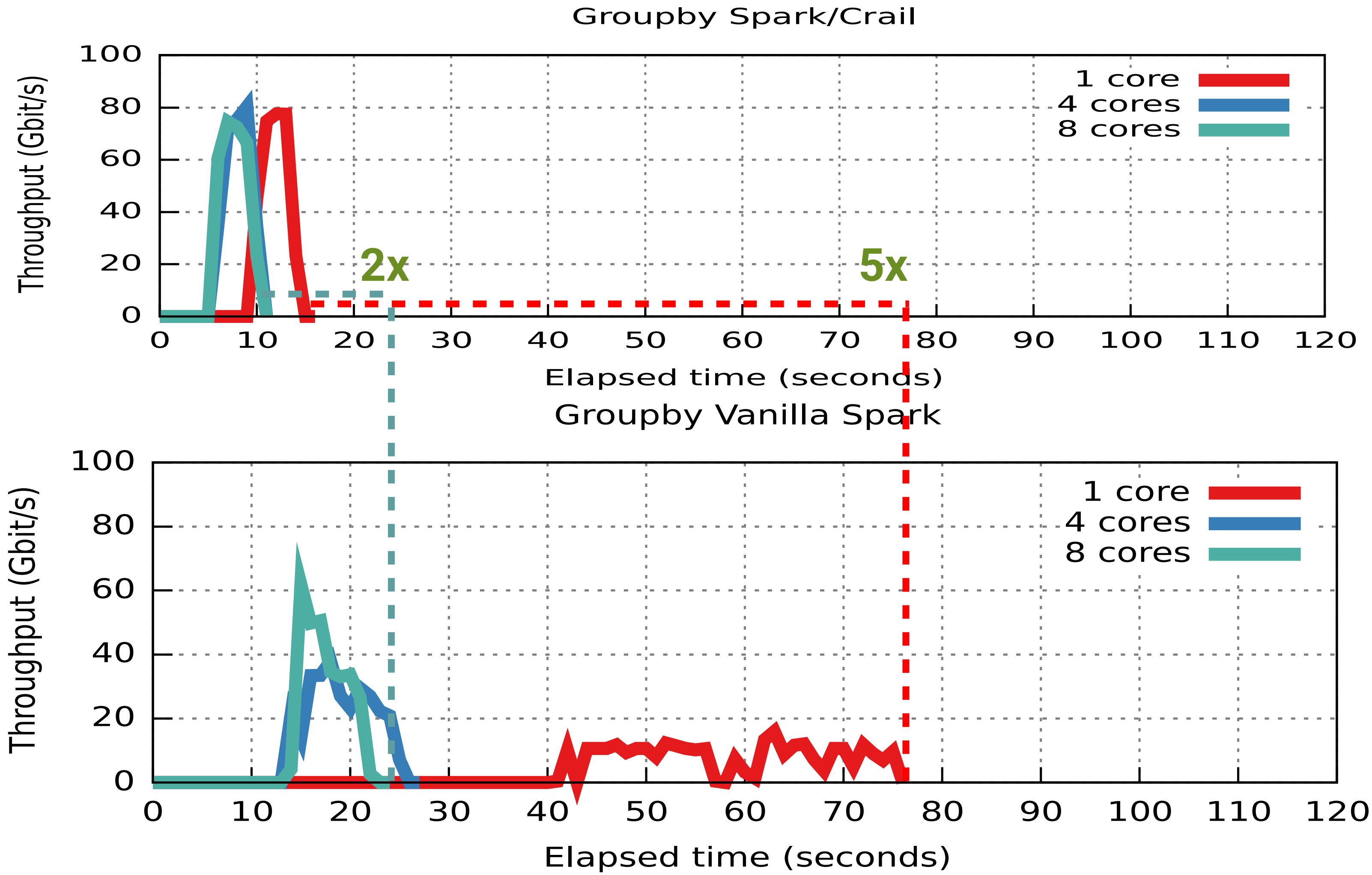


Crail Workload - Spark - Broadcast



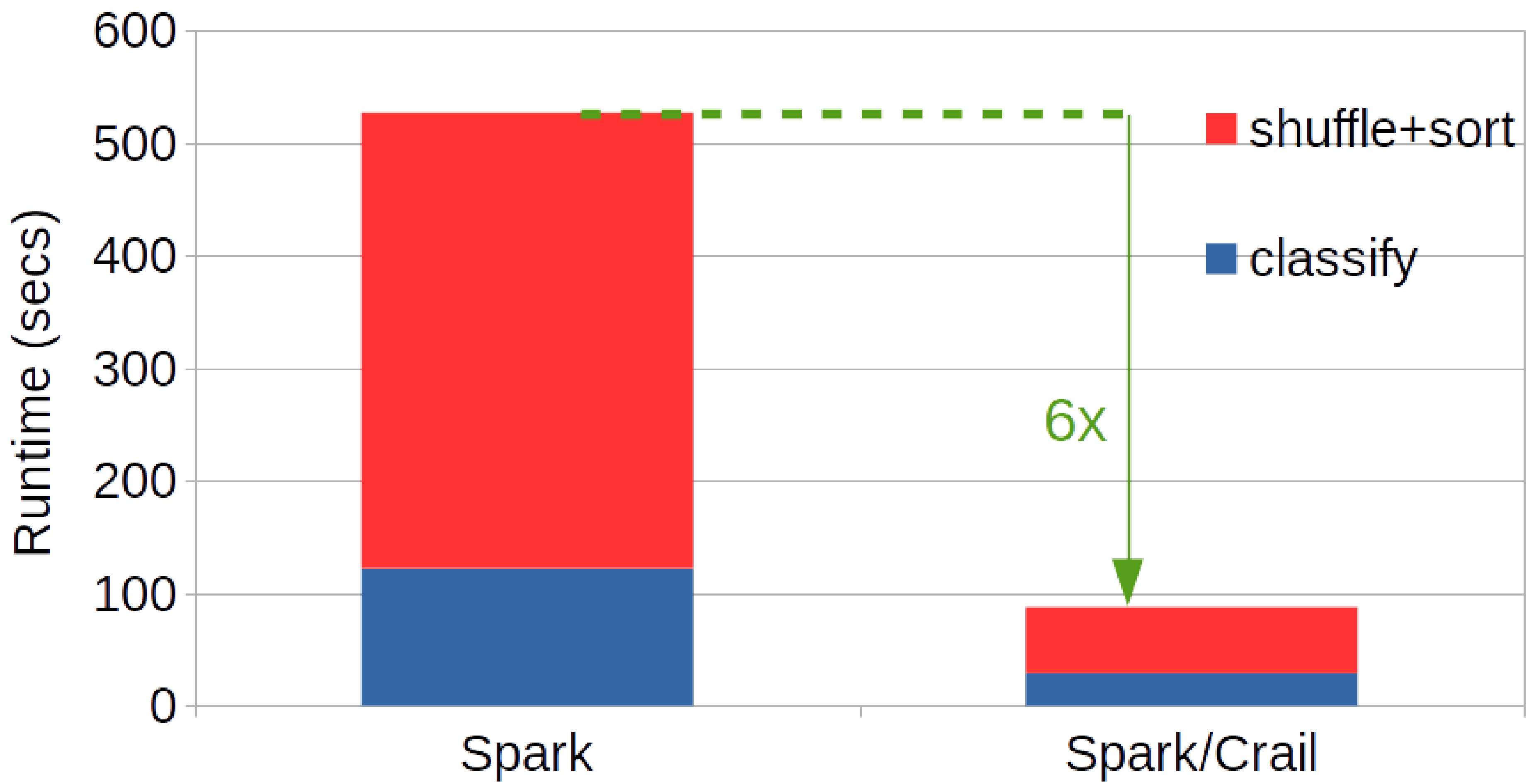
Crail offers 10-100x performance gains for Spark Broadcasts

Crail Workload - Spark - GroupBy



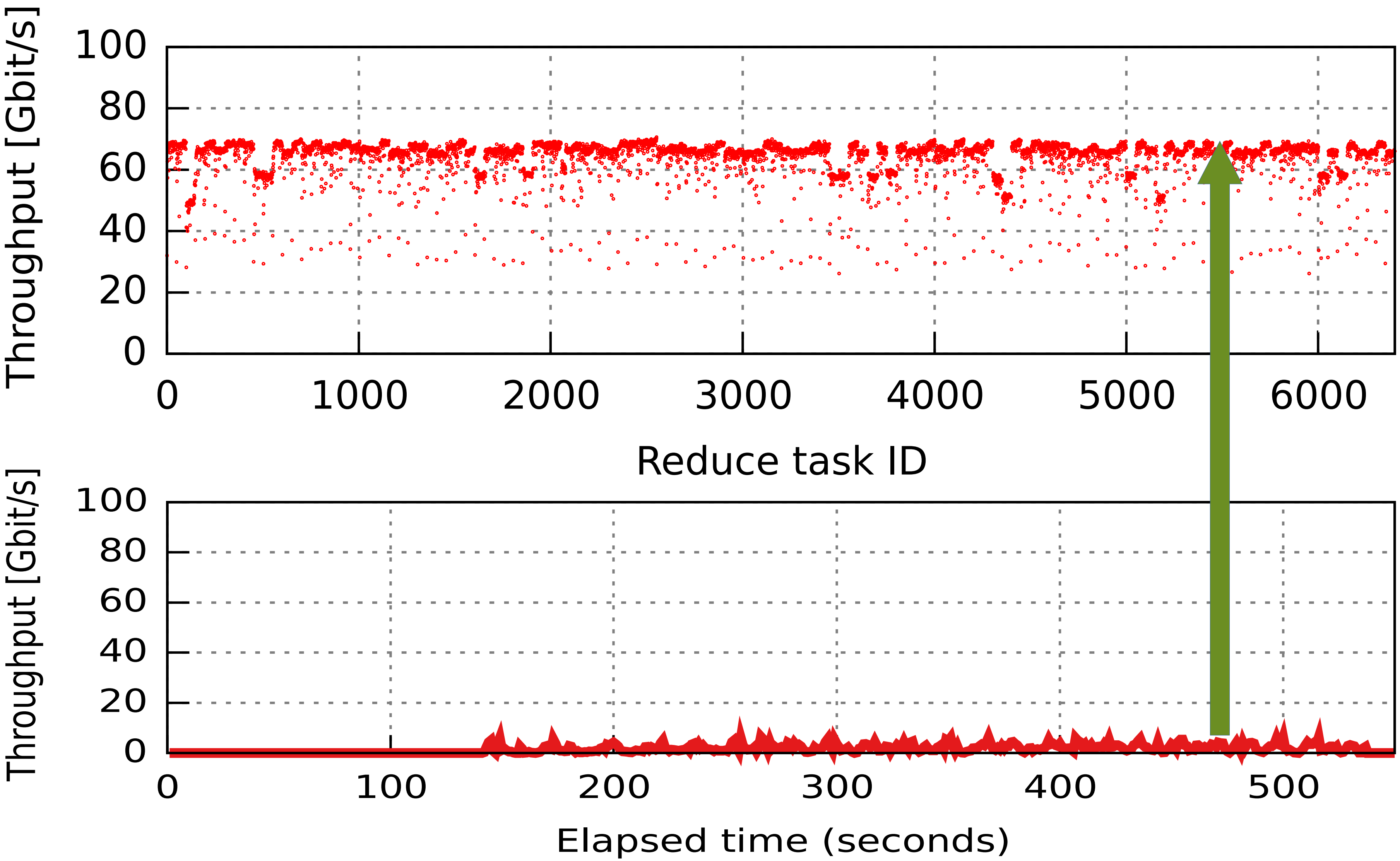
Crail offers 2-5x performance gains for Spark Shuffle

Crail Workload - Spark - TeraSort

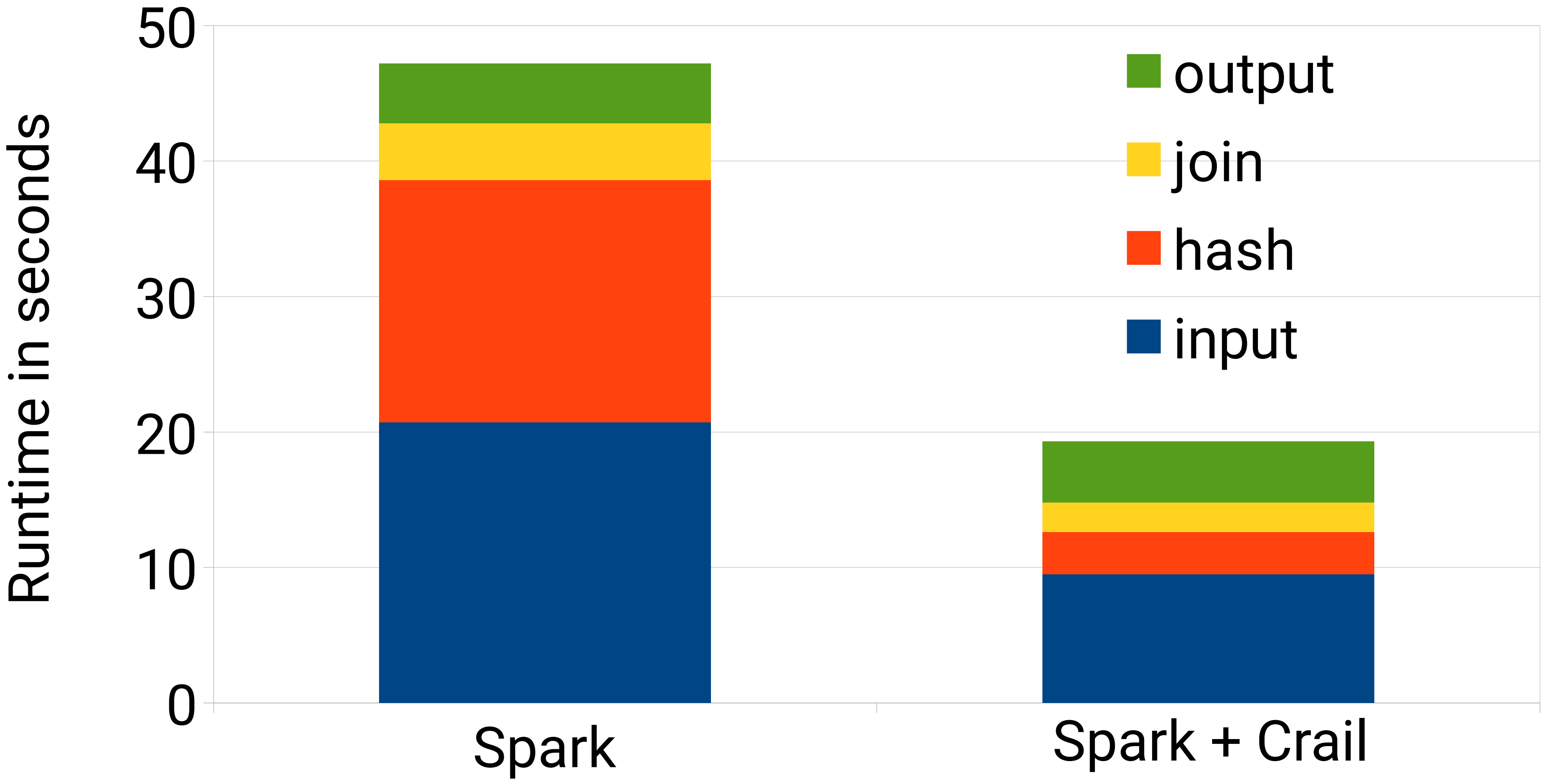


Crail offers 6x performance gains for Spark TeraSort [7]

Crail Workload - Spark - TeraSort

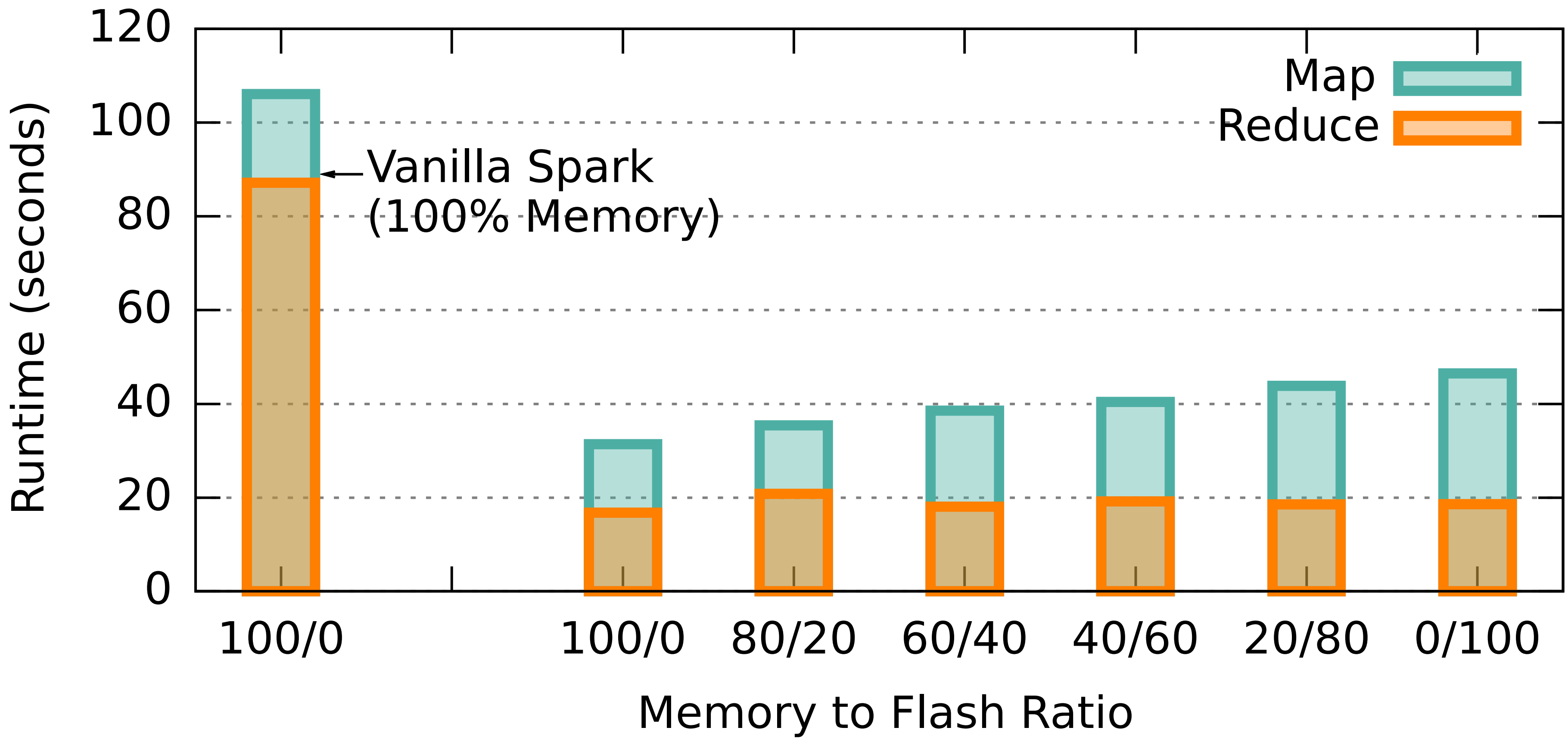


Crail Workload - Spark - EquiJoin



Crail offers 2x performance gains for SQL JOIN

Crail Store – Flash Disaggregation



Crail offers the possibility for storage dis-aggregation

Current Status and Future Plans

- Apache Incubator project since November, 2017
  - First source release (tag: 0db64fd), a few weeks ago
- Plenty of opportunities
  - Multiple languages support (C++ (WiP), Python, Rust, _your_fav_language_)
  - Multiple frameworks integration (Flink, Hadoop, λ compute, TensorFlow, SnapML[5], _your_fav_framework_here_)
  - Multiple datanode (network and storage) integration
  - Automated testing and packaging framework
  - Deploying in the cloud/containerization
  - JVM Optimizations
  - And all the fun stuff that you know and love about Apache projects !

Thanks to

Patrick Stuedi, Jonas Pfefferle, Michael Kaufmann, Adrian Schuepbach, Bernard Metzler

Thank you !

crail

See you all at the mailing list :)

- Website: http://crail.incubator.apache.org/
- Blog: http://crail.incubator.apache.org/blog/
- Mailings list: dev@crail.incubator.apache.org
- JIRA: https://issues.apache.org/jira/browse/CRAIL
- Slack: https://the-asf.slack.com/messages/C8VDLDWMV

References

[1] DiSNI: Direct Storage and Networking Interface, https://github.com/zrlio/disni

[2] jNVMf: A NVMe over Fabrics library for Java, https://github.com/zrlio/jNVMf

[3] "*Crail: A High-Performance I/O Architecture for Distributed Data Processing*", in the IEEE Bulletin of the Technical Committee on Data Engineering, Special Issue on Distributed Data Management with RDMA, Volume 40, pages 40-52, March, 2017.
http://sites.computer.org/debull/A17mar/p38.pdf

[4] Crail I/O accleration plugins for Apache Spark, https://github.com/zrlio/crail-spark-io

[5] Snap machine learning (SnapML), https://www.zurich.ibm.com/snapml/

[6] Apache Crail (Incubating) performance blogs

- Part I: DRAM, http://crail.incubator.apache.org/blog/2017/08/crail-memory.html
- Part II: NVMeF, http://crail.incubator.apache.org/blog/2017/08/crail-nvme-fabrics-v1.html
- Part III: Metadata, http://crail.incubator.apache.org/blog/2017/11/crail-metadata.html

[7] Sorting on a 100Gbit/s Cluster using Spark/Crail, http://crail.incubator.apache.org/blog/2017/01/sorting.html

Notice

IBM is a trademark of International Business Machines Corporation, registered in many jurisdictions worldwide. Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Java and all Java- based trade-marks and logos are trademarks or registered trademarks of Oracle and/or its affiliates Other products and service names might be trademarks of IBM or other companies.