# Experiences Running and Optimizing the Berkeley Data Analytics Stack on Cray Platforms

**Kristyn J. Maschhoff and Michael F. Ringenburg**

**Cray Inc.**

**CUG 2015**

# Legal Disclaimer

*Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.*

*Cray Inc. may make changes to specifications and product descriptions at any time, without notice.*

*All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.*

*Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.*

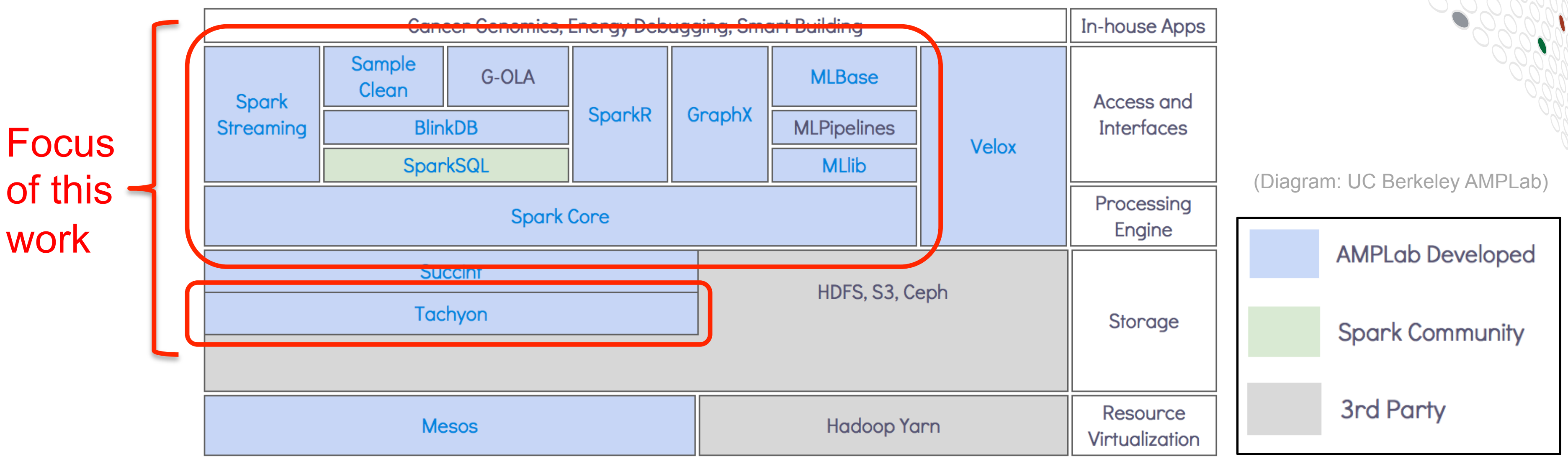
*Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.*

*Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.*

*Cray and Sonexion are registered trademarks of Cray Inc. in the United States and other countries, and Cray XC30, Cray CS300, Cray XK7, Cray XE6, Cray Linux Environment, Cray XE6m, Cray XE6m-200, Cray XT6, Cray XT5, Cray XT4, Cray SHMEM, CrayPat, NodeKARE and Urika are registered trademarks of Cray Inc.*

*Other names and brands may be claimed as the property of others. Other product and service names mentioned herein are the trademarks of their respective owners.*

# Berkeley Data Analytics Stack (BDAS)

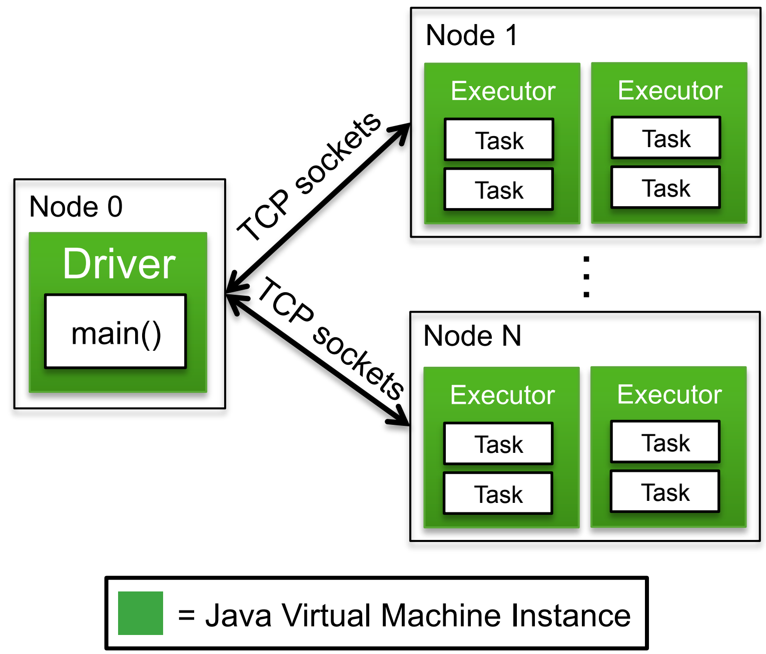



(Diagram: UC Berkeley AMPLab)

- **Spark in-memory analytics framework**
  - Includes modules for graph analysis, SQL, machine learning, and streaming
- **Tachyon distributed in-memory file system**
- **Mesos cluster manager**

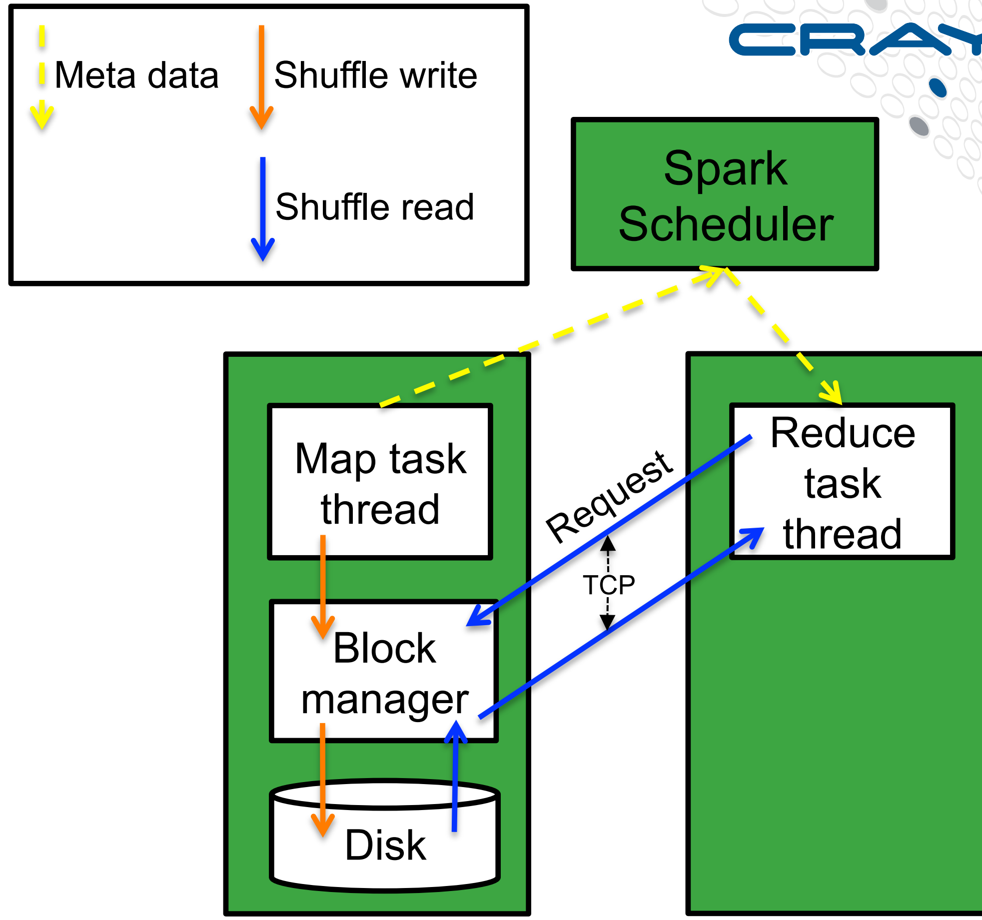
# Spark Execution Model

- **Master-slave parallelism**
- **Driver (master)**
  - Executes main
  - Distributes RDDs & tasks to executors
- **Resilient Distributed Dataset (RDD)**
  - Spark's primary data abstraction
  - Partitioned amongst executors
  - Fault-tolerant via lineage
- **Executors (slaves)**
  - Lazily execute tasks (operations on partitions of the RDD)
  - Global all-to-all shuffles for data exchange



# Spark Shuffle

- **All data exchanges between executors implemented via *shuffle***
  - Senders ("mappers") send data to block managers; block managers write to disks, tell scheduler *how much* destined for each reducer
  - Barrier until all mappers complete shuffle writes
  - Receivers ("reducers") request data from block managers *that have data for them*; block managers read and send



# Spark Programming Model: Example

Create array of {1, 2, …, 1,000,000}

Partition array into a 40-partition RDD distributed across executor nodes.

(Can also create from file.)

Spark *transformation* (modify data in RDDs)

Spark *action* (return result to driver)

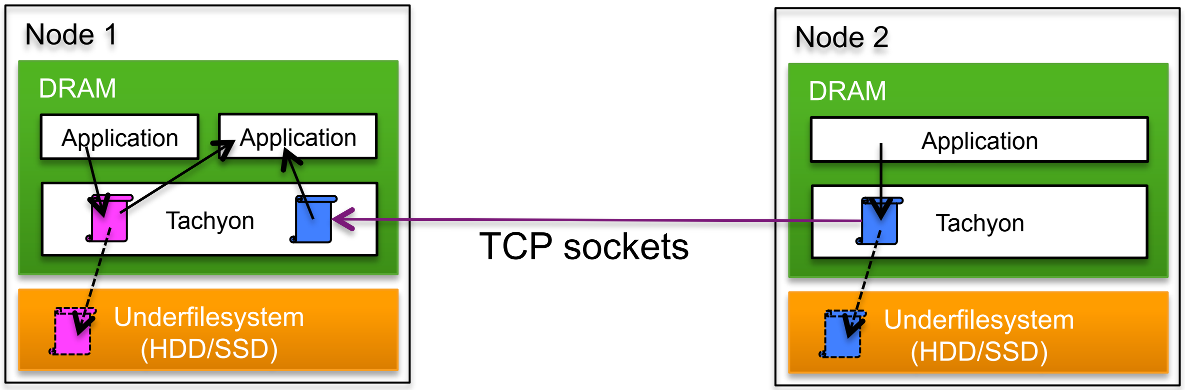
```
val arr1M = Array.range(1,1000001)
val rdd1M = sc.parallelize(arr1M, 40)
val evens = rdd1M.filter(
              a => (a%2) == 0
            )
evens.take(5)

>>> Array[Int] = Array(2, 4, 6, 8, 10)
```

compute

Lazy Evaluation: No computation until result requested

# Tachyon




- **Distributed in-memory filesystem**
  - HDD/SSD I/O replaced with DRAM loads and stores
- **Fault tolerance via:**
  - Asynchronous checkpoints to (persistent) underfilesystem
  - Persistent *lineage* tracking

# Spark and Tachyon on Cray Systems

- **This paper reports our experiences running and optimizing BDAS on three Cray systems**
  - Urika-XA Exterme Analytics Platform
    - 48 dual socket nodes, 16-core Haswell, FDR Infiniband, 800 GB SSD and 1TB HDD on every node, 128 GB DRAM/node
    - Cloudera Distribution of Hadoop 5.3, w/ Spark 1.2
  - Prototype Aries-based system with node-local SSDs
    - 43 dual socket nodes, 12-core Haswell, Cray Aries, 800 GB SSD and 1TB HDD on every node, 128 GB DRAM/node
    - Spark 1.3 and Tachyon 0.6.1 on top of CentOS 6.4
  - XC 40
    - Used 43 nodes, dual 16-core Haswell, Cray Aries interconnect, 128 GB DRAM/node
    - Spark 1.3 in Cluster Compatibility Mode (CCM)

# Configuring Spark for Cray Systems

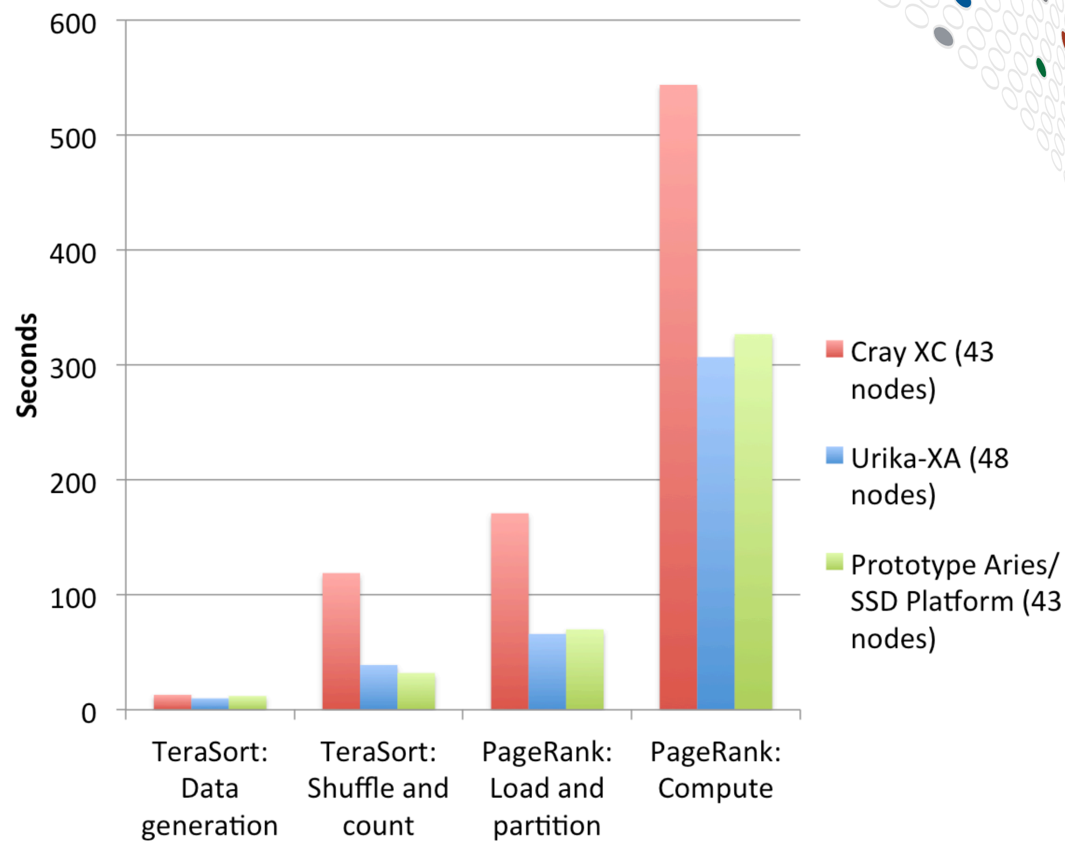
- **Spark Shuffle: responsible for data movement between executors**
  - File I/O is often shuffle bottleneck
  - **Sort-based shuffle**: consolidates intermediate files; friendlier to OS cache
  - **Urika-XA, Prototype Aries system**: moved shuffle files to local SSDs
  - **XC systems**: placed shuffle files in local RAM disk
    - Has tendency to fill RAM, so allocated secondary shuffle directory on Lustre
- **Studying additional configs related to network capabilities**
  - Default parameters tuned for commodity interconnects: willing to spend a lot of compute time to save on network traffic
  - Does this make sense with a more capable interconnect?
  - Recent research (Ousterhout et al, NSDI '15 in May) indicates network tuning may have gone to far – compute now the bottleneck…

# Configuring Spark Memory Usage

- **Need to balance memory usage between Spark, Virtual Machines/Interpreters, OS file buffer**
  - Extra executor memory minimizes spills to disk
  - Leaving "slack" in Java heap => less garbage collection overhead
  - Extra memory not used by applications => larger OS file buffer (improves shuffle performance)
  - On XC, more RAM disk space improves shuffles
  - Best performance in our tests: typically ~50% of total memory to executors

# Spark Results

- **Local SSDs provide large benefit**
  - Aries prototype and Urika-XA SSDs vs XC RAM disk
  - Preventing RAM disk exhaustion requires backing with Lustre
- **Recent results (post-paper) show we can cut another 25% by eliminating compresssion**



# Tachyon Results

| Dataset | Size | Lustre to GraphRDD | Tachyon to GraphRDD | Tachyon Load Speedup |
|---|---|---|---|---|
| LiveJournal | 1.0 GB | 13.8 seconds | 5.4 seconds | 2.6x |
| Twitter | 24.3 GB | 41.1 seconds | 19.4 seconds | 2.1x |

- **Compared loading GraphX edgelist file from Lustre vs Tachyon**
  - Flushed OS file caches between runs
- **At least 2x speedup from Tachyon**

# Potential Future Optimizations

- **Replace TCP sockets with native Aries communication**
  - **High-performance Big Data project at Ohio State**: Hadoop, HDFS, and Spark via RDMA over Infiniband
  - **Unstructured Data Acellerator (UDA) plugin (Mellanox, Auburn University)**: Hadoop MapReduce Shuffle over Infiniband
- **Explore causes of RAM disk exhaustion on XC**
  - Appears to fill up quicker than should
  - Currently round robin betweek RAM/Lustre … bias towards RAM?
  - Investigating Spark code w/ AMPLab assistance
  - Or, move shuffle files to DataWarp

# Potential Future Optimizations

- **Integrate optimized libraries and engines**
  - Linear algebra common in MLLib (Spark machine learning) – AMPLab sped up perfomance by swapping in optimized linear algebra libraries
  - Cray Graph Engine (see CUG paper, talk earlier this week) outperforms GraphX algorithms by 10x
  - Investigate calling Cray libraries and integerating with CGE
- **Continue investigation of Compute/Network configuration tradeoffs**
  - Compression
  - Locality wait
  - Speculation
  - Max MB in flight

# Summary

- **Paper describes our experiences running Spark and Tachyon across a variety of Cray platforms**
- **Investigated configurations, tuning, and potential optimizations**
  - Network/compute tradeoffs
  - Shuffle improvements
  - Tigher integration of Cray libraries, engines
- **Questions? Contact the authors (mikeri@cray.com, kristyn@cray.com), or Venkat Krishnamurthy <venkat@cray.com>.**