Characterizing the Performance of Analytics Workloads on the Cray XC40

Cray: Michael Ringenburg, Shuxia Zhang, Kristyn Maschhoff, Bill Sparks
NERSC: Evan Racah, Prabhat

SCALABILITY

London | May 8-12

Agenda

- **Last year at CUG: Showed how to run common open source analytics frameworks on XC systems**
- **Today: How do we understand, monitor, tune performance on XC40?**
  - Focus on Apache Spark framework – more flexibility, better performace, increasing adoption relative to Hadoop
    - SyncSort Survey: 70% Spark interest vs 55% Hadoop.
- **Look at a couple use cases/techniques, plus a networking analysis**
  - Bottom-up: mining system metrics data from HiBench with `collectl`
  - Top down: application log analysis of CX matrix decompostion in Spark
  - TCP networking performance on XC
- **In the paper: additional details, plus suggested optimizations**

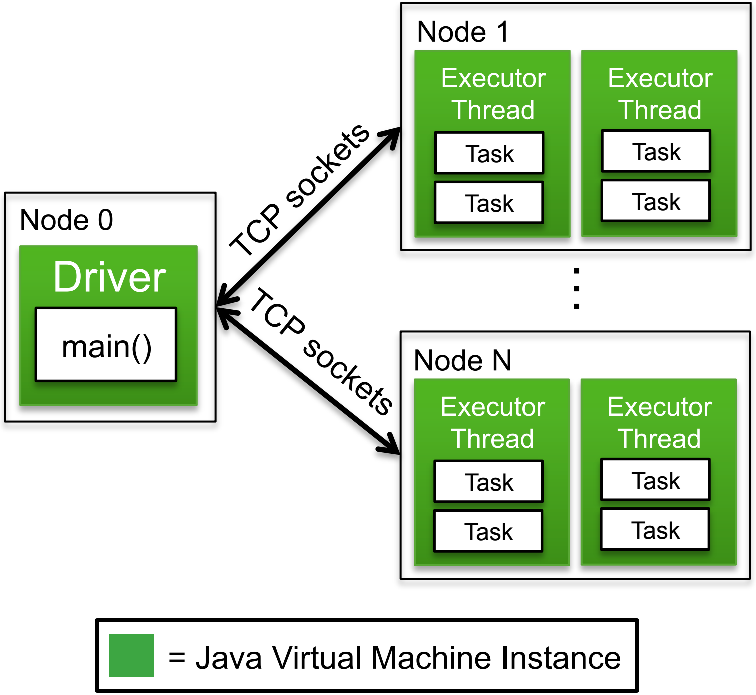
Copyright 2016 Cray Inc.

Spark Background: Execution Model

- **Driver is the "master": execute main, distribute work , collect results**
- **Executors are the "workers": execute parallel work across partitions of the data**
- **Computations lazily evaluated – nothing happens until result required at driver:**

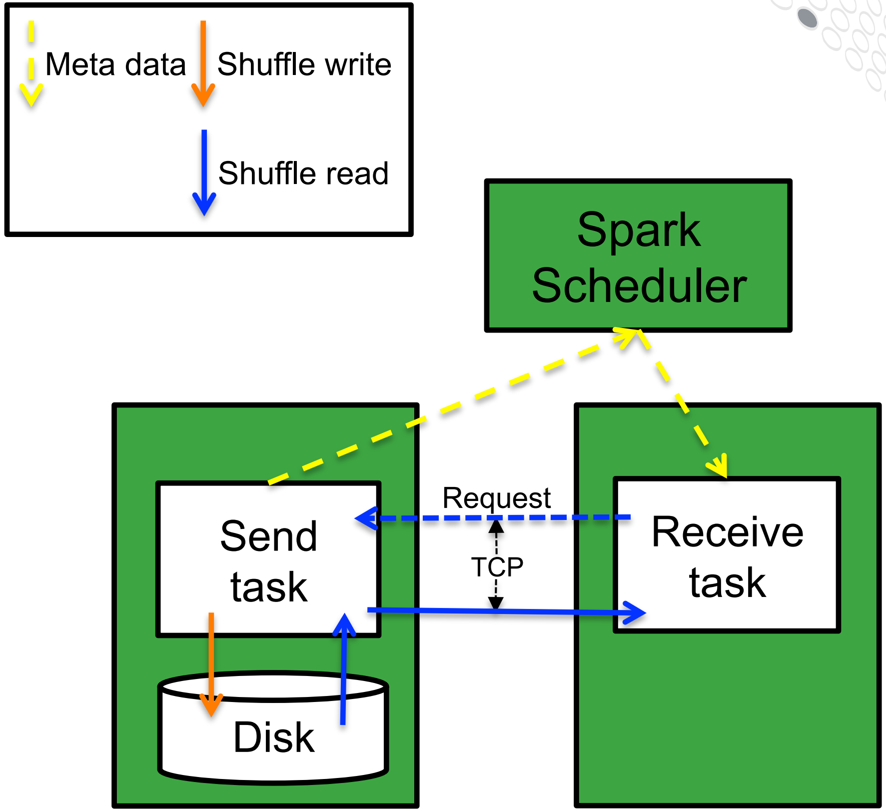
```
val lens = file.map(l => (l.length,l))
val sorted = lens.sortByKey()
sorted.collect()  // execution starts HERE
```

- **Jobs, Stages, and Tasks:**
  - Job: Computation that returns a result to driver
  - Stage: Unit of work that can be executed without communication. Jobs with internode communication requirements have multiple stages.
  - Between job stages: barrier, global all-to-all shuffle
  - Task: The computation of a stage on a single partition



Spark Background: Shuffle

- **Communication between executors implemented via *shuffles***
  - Senders send data to block managers; block managers write to disks, tell scheduler how much destined for each receiver
  - Barrier until all senders complete shuffle writes
  - Receivers request data; block managers read and send



Copyright 2016 Cray Inc.

Shuffle on the XC40

- **Spark assumes distributed cluster, with local persistent storage on each node for shuffle files (also for spilling RDDs).**
- **Not present on XC systems. Options:**
  - Global Lustre file system: Many small files, and file opens/closes = high metadata overheads that dominate performance of shuffles.
  - DRAM-based tmpfs: Much faster, but storage limited to 50% of memory on node. Works for many workloads, but can run into memory bottlenecks.
  - Hybrid: Use both. Better performance than pure Lustre.
  - Loopback filesystems (see earlier presentation in Session 7A): Each node create a filesystem within a single Lustre file. Managed locally. Eliminates MDS overheads, coherency issues.

Copyright 2016 Cray Inc.

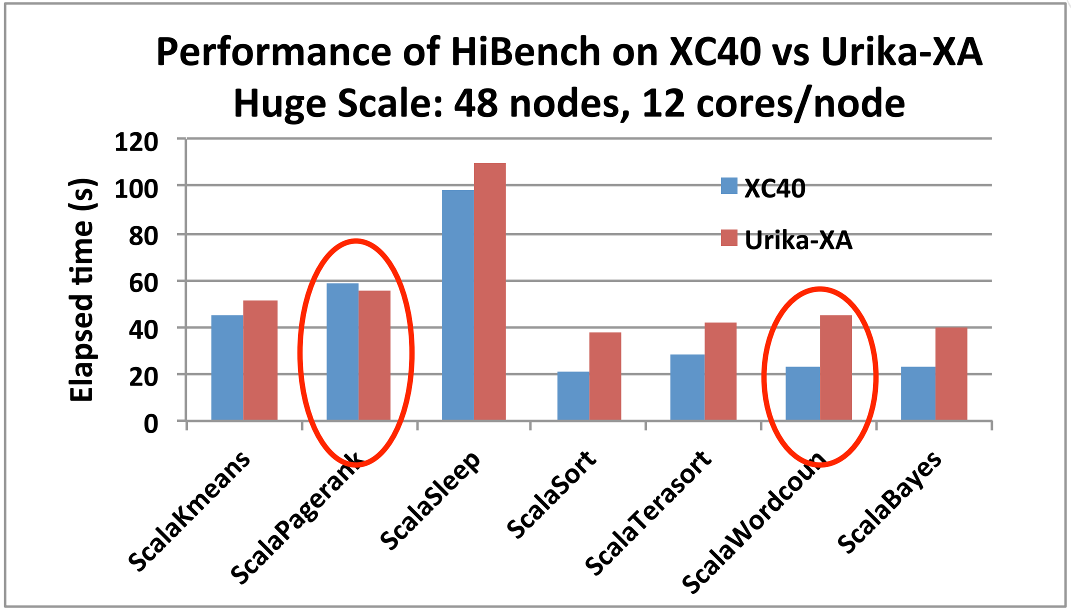
Our Analysis Approaches

- **Collectl**
  - Commonly used for collecting compute node system metrics for HPC jobs
  - Used a 1 second sampling rate
    - Negligible overheads next to overheads of analytics frameworks – found to have no impact on completion time of our workloads
    - Accurate: tested by comparing aggregated Lustre metrics with input and output data set sizes, saw less than 1% variation
  - Used R+pdbMPI to analyze, plot results
- **Spark event logs**
  - Track start and end times of jobs, tasks, stages
  - Collect application level metrics for each task (GC time, serialization time, shuffle read/write, etc)
  - Can view in Spark History Server, or parse with scripts
- **TCP network performance analysis with iperf3, tcpdump**

Copyright 2016 Cray Inc.

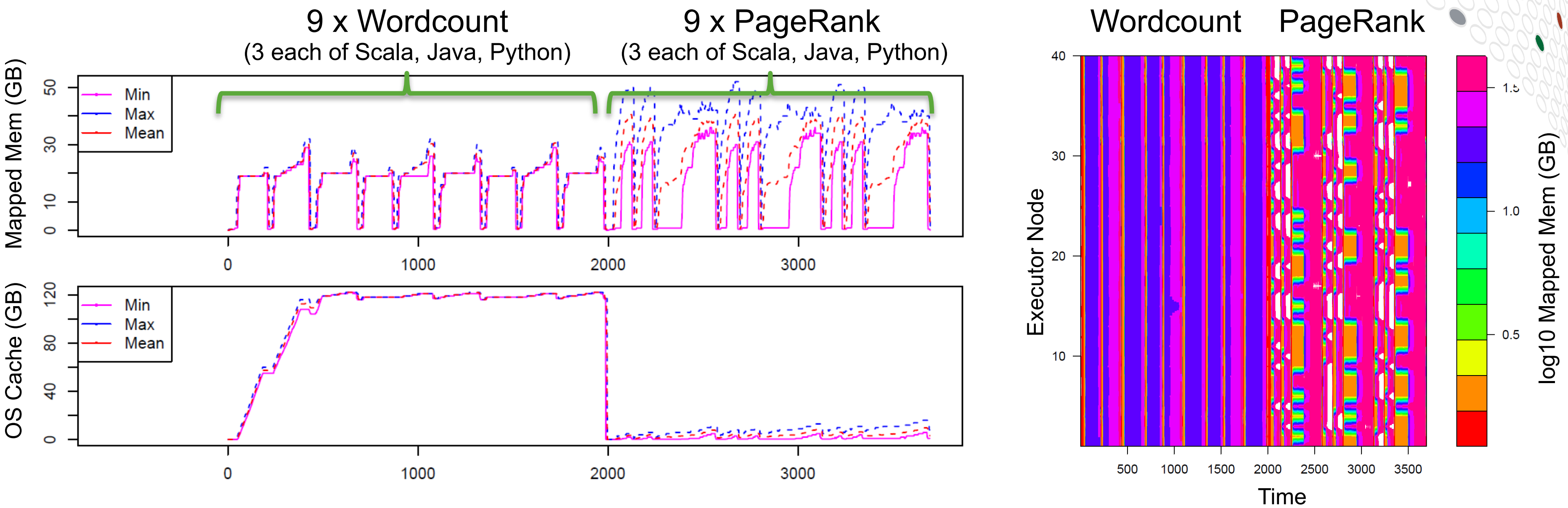
Case Study 1: HiBench Analysis

- **Intel HiBench**
  - Originally MapReduce, Spark added in version 4
  - We selected common Spark workloads without Hive dependencies
- **Compared performance with Urika XA system**
  - XA: FDR Infiniband, XC40: Aries
  - Both: 32 core Haswell nodes
  - XA: 128 GB/node, XC40: 256 GB/node (problems fit in memory on both)
- **Similar performace on Kmeans, PageRank, Sleep**
- **XC40 faster Sort, TeraSort, Wordcount, Bayes**
- **Let's examine…**



Copyright 2016 Cray Inc.

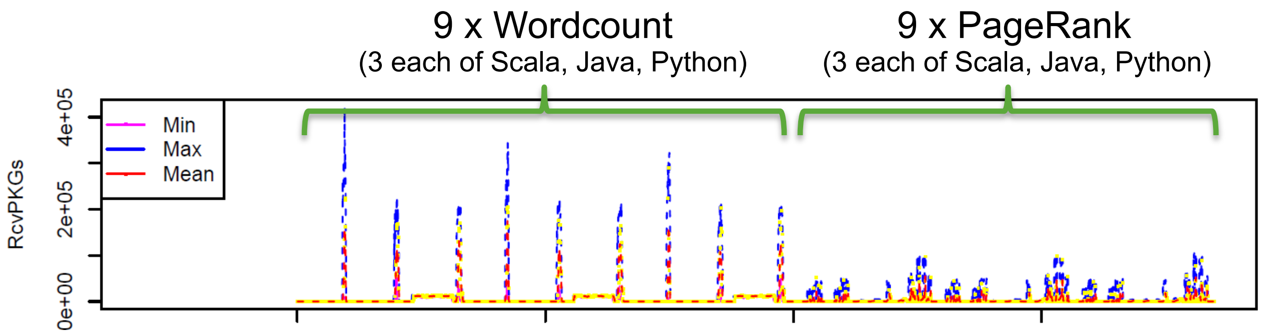
Collectl: Examining Memory Usage



- **PageRank much larger variation between executors in memory usage**
  - Points to variation in data set/# links per page
- **Wordcount much higher OS file cache usage**
  - Spark uses file system for spills and shuffle data

Copyright 2016 Cray Inc.

Collectl Example: Understanding Memory Usage



- **Examing TCP traffic points to reason for larger OS cache usage: Larger shuffles**
- **Variability of PageRank indicates bottleneck due to stragglers/imbalance**
- **Wordcount has larger shuffles, and much less variability, indicating potential bottleneck in data movement**
  - Takes better advantage of interconnect

Copyright 2016 Cray Inc.

Benefits of Collectl-Based Profiling

- **Spark event log/history server can't give this level of detail**
  - Only shows static size of persisted RDDs/DataFrames
  - Only shows application memory usage (no OS)
  - Only shows total shuffle traffic per stage
  - Etc…
- **Collectl-based analysis allows you to view all system metrics, and how they change over time**

Copyright 2016 Cray Inc.

Case Study 2: CX for MSI

- **CX matrix decomposition applied to mass spectrometry data from bioimaging (Spark implementation from NERSC and AMPLab)**
  - Analysis of 1 TB Mass Spectrometry (MSI) dataset
    - Matrix with columns for each spatial location
  - Method: dimensionality reductions via CX factorization
    - Approximately factors *m x n* matrix A into *m x k* matrix C *and k x n* matrix X, where the *k* columns of C are drawn from A. The "rank" k is typically much less than n
    - Goal is to select the *k* columns of A such that CX is as close as possible to A.
    - These columns best "explain" the data in the matrix
- **These types of matrix decomposition operations are also common in Machine Learning applications**

Copyright 2016 Cray Inc.

Comparison Platforms

Platform	Total Cores (Haswell)	Core Frequency	Interconnect	DRAM	SSDs/ node
Amazon EC2 `r3.8xlarge`	960 (30 nodes x 32 per-node)	2.5 GHz	10 Gigabit Ethernet	256 GB	2 x 320 GB
Cray XC40	960 (30 nodes x 32 per-node)	2.3 GHz	Aries	256 GB	None
Athena early prototype	960 (40 nodes x 24 per-node)	2.5 GHx	Aries	128 GB	1 x 800 GB

Platform	Total Runtime
Amazon EC2	24.0 min
XC40 w/ tmpfs & Lustre	23.1 min
XC40 w/ tmpfs	18.1 min
Athena early prototype	15.2 min

- **Two options for XC40 scratch space: DRAM tmpfs, or tmpfs & Lustre blend**

Copyright 2016 Cray Inc.

Performance Analysis via Event Logs

Platform	Total Runtime	Load Time	Time per iteration	Average Local Task	Average Aggregation Task	Average Network Wait
Amazon EC2	24.0 min	**92 sec**	161 sec	4.4 sec	**27.1 sec**	**21.7 sec**
Cray XC40 w/ tmpfs & Lustre	23.1 min	**139 sec**	125 sec	3.5 sec (max: 12!)	**6.8 sec**	**1.1 sec**
Cray XC40 w/ tmpfs	18.1 min	**137 sec**	94 sec	3.0 sec	**7.3 sec**	**1.5 sec**
Athena prototype	15.2 min	**53 sec**	92 sec	2.8 sec	**9.9 sec**	**2.7 sec**

- **Workload:** Load MSI dataset; 5 iterations, each with local stage (compute local sums) and aggregation stage
- **Observations:**
  - Load times faster on machines with local SSD storage
  - Aggregation stage tasks (shuffle read) much faster on Aries-based systems
  - Network wait on XC40 was lower than Athena, due to better peak TCP bandwidth
  - All tmpfs, or fast local SSDs sped up local task time (shuffle write)
  - Mixing Lustre and tmpfs adds long tail to shuffle write time distribution, creating stragglers that slow iterations

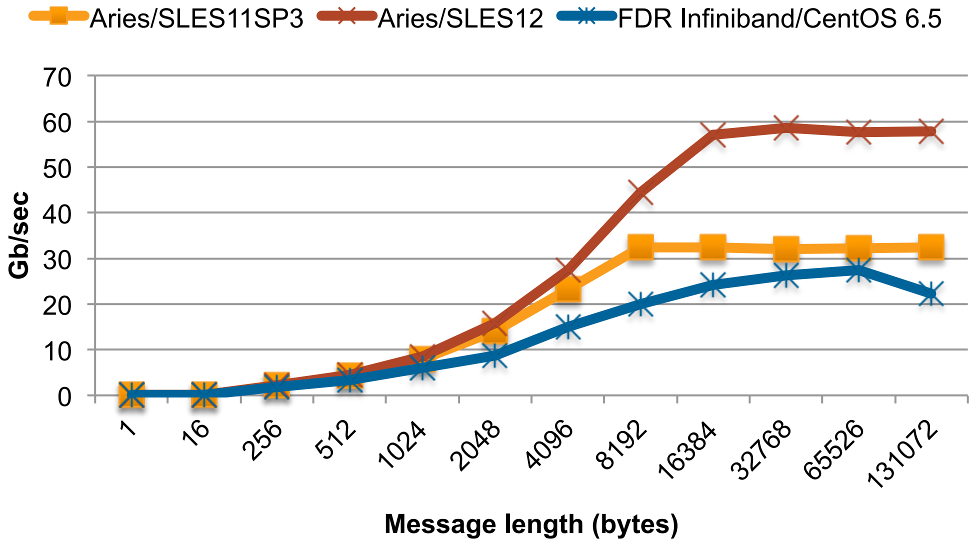
Copyright 2016 Cray Inc.

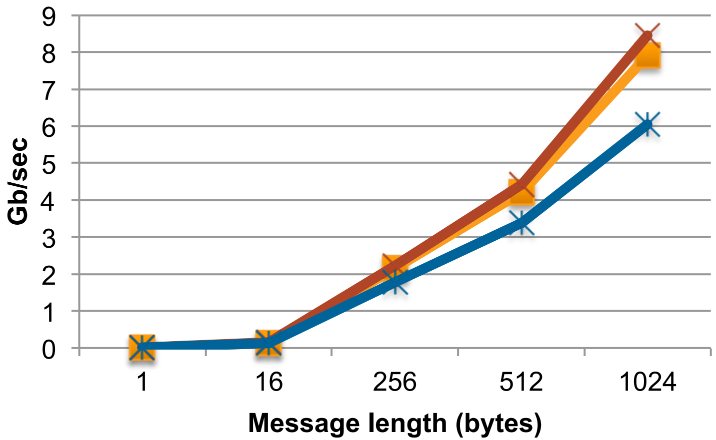
Advantages of Log Analysis

- **Application level view**
  - What tasks are bottlenecks
  - Where applications are spending their time (e.g., garbage collection, serialization, compression, waiting at barriers, etc)
- **Tools integrated with the Spark/Hadoop/etc ecosystem**
  - WebUI
  - Visualizations that have application-level information (stages, operations, etc)
  - User familiarity

Copyright 2016 Cray Inc.

Analysis of TCP Bandwidth w/ iperf3

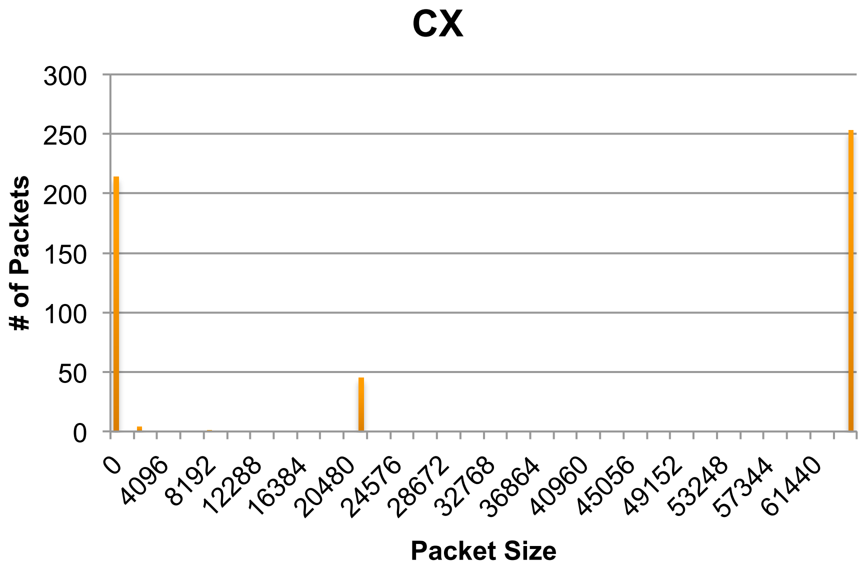


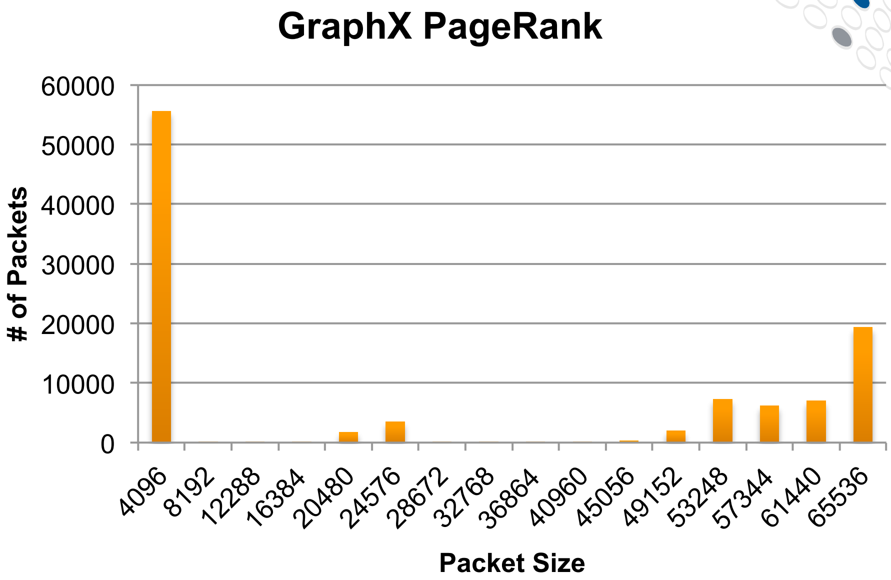


- **Communication in open source analytics frameworks is typically over TCP (for portability)**
- **Aries performs well, especially with new kernel. Peak performance hit at ~ 8K-16K.**

Copyright 2016 Cray Inc.

TCP Packet Size Distribution





- **Picked two shuffle senstive applications – CX and GraphX PageRank**
  - CX is more uniform (sending matrix columns) than PageRank (variable sized edge lists)
  - Both have a number of large data packets
    - CX: 58% at least 21KB
    - PageRank: 46% at least 16KB

Copyright 2016 Cray Inc.

Summary

- **Described multiple approaches to understand analytics framework performance on XC40**
  - System metrics data from `collectl`
  - Application log analysis
  - Network performance (TCP)
- **Looked at two benchmarks**
  - HiBench suite
  - CX (NERSC implementation)
- **Paper discusses performance improvement possibilities**

Copyright 2016 Cray Inc.

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.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, and URIKA. The following are trademarks of Cray Inc.: APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, REVEAL, THREADSTORM. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.

Copyright 2016 Cray Inc.

Q&A

Michael Ringenburg
mikeri@cray.com

Copyright 2016 Cray Inc.