



Significantly Speed up real world big data Applications using Apache Spark

Mingfei Shi(mingfei.shi@intel.com)
Grace Huang (jie.huang@intel.com)
Intel/SSG/Big Data Technology



Agenda

- Who are we ?
- Case study and optimization experience
 - Machine Learning(Graph analysis)
 - Batch style analysis
 - Batch style OLAP
 - Streaming & Interactive OLAP
- Dew – Assistant for workload tuning on Spark
- Lessons learned & Summary

Who are we ?

- Intel SSG Big data technology team
- Focus on Spark development and promotion in industry
 - Long history in Spark with AMPLab and community
 - Key contributions to grow it and its ecosystem
 - Among Top contributors
- 10 active contributors on Spark related projects

Spark is Sparkling

- Spark is skyrocketed
 -  +  Spark co-locates in data center
- Intel partnering with several large organizations/websites in China since 2012
 - Building real-world big data analytic applications using Spark stack

Building next-gen big data analytics

- Advanced ML and Graph Analysis
 - Relationship analysis
 - Similarity measure
 - Community detection
- Complex / Interactive Analysis
 - Batch style OLAP Analysis
 - Interactive/ad-hoc OLAP Analysis
- Real-time* Stream processing
 - Log analysis

Experience from partnership

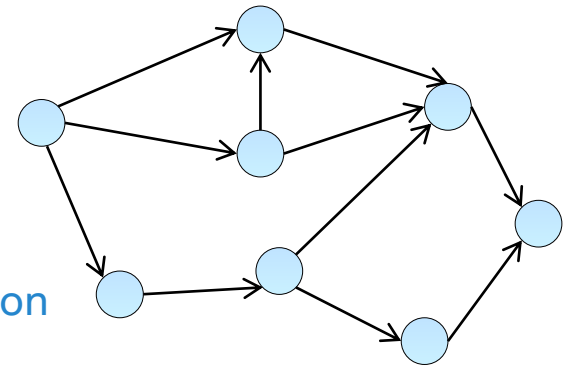
- Significant speedup in real-world applications
 - x5-100 performance gains versus to Hadoop MR
- Easy of use on a common deployment
 - All in one platform
 - Interactive programming interface
 - Scala like API
- Spark application can perform even better through optimization

Agenda

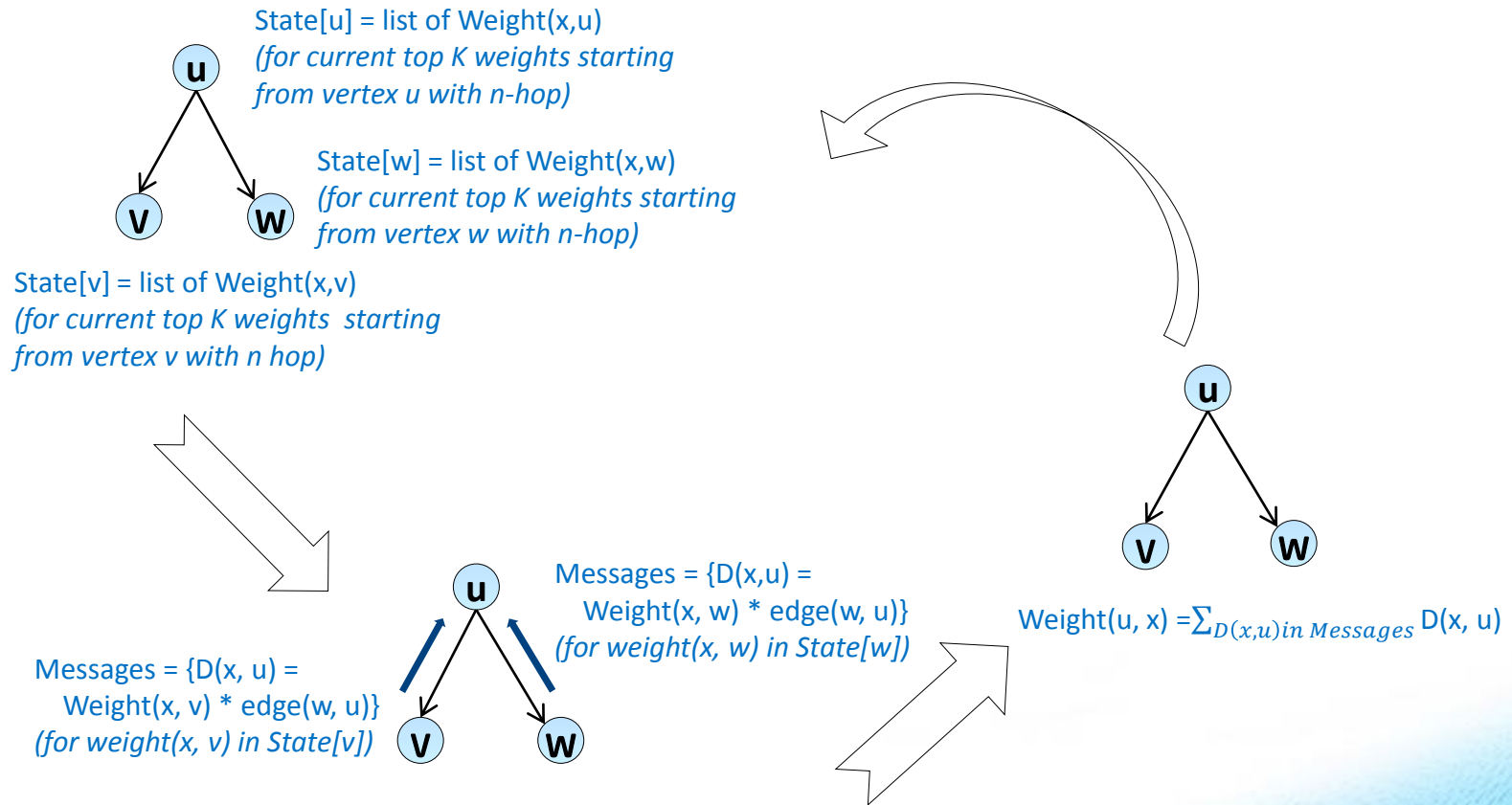
- Who are we ?
- Case study and optimization experience
 - Machine Learning(Graph analysis)
 - Batch style analysis
 - Batch style OLAP
 - Streaming & Interactive OLAP
- Dew – Assistant for workload tuning on Spark
- Lessons learned & Summary

Graph Analysis – N degree

- Computing associations between two vertices that are *n-hop* away
 - $Weight_1(u, v) = edge(u, v) \in (0, 1)$
 - $Weight_n(u, v) = \sum_{x \rightarrow v} Weight_{n-1}(u, x) * Weight_1(x, v)$
- A Graph Analysis case
 - E.g., friends of friend in social network
 - E.g., relationship between videos for recommendation
- Graph-parallel implementation
 - Bagel (Pregel on Spark)
 - Speedup from 20 minutes to 2 minutes compared with customer's original MR implementation

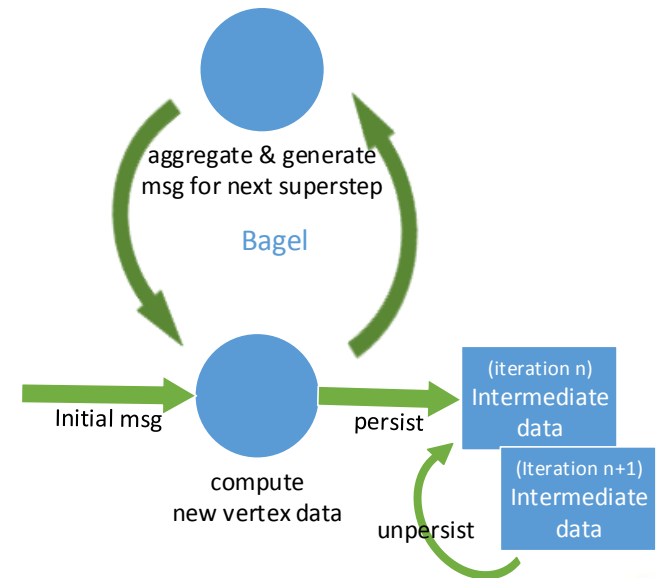


Work model



Optimization- Free memory space timely

- Bagel work flow
 - Cache intermediate data of each iteration
- The present iteration only depends on its previous step
 - I.e., Intermediate data in iteration N (RDD[n]) is only used in iteration N + 1
 - Memory space is continuously increased in Bagel
- Free those obsolete intermediate data not be used anymore
 - *Un-persist intermediate data (RDD[n])* after iteration N + 1 is done
 - SPARK-2661 solve the issue



Memory usage optimization

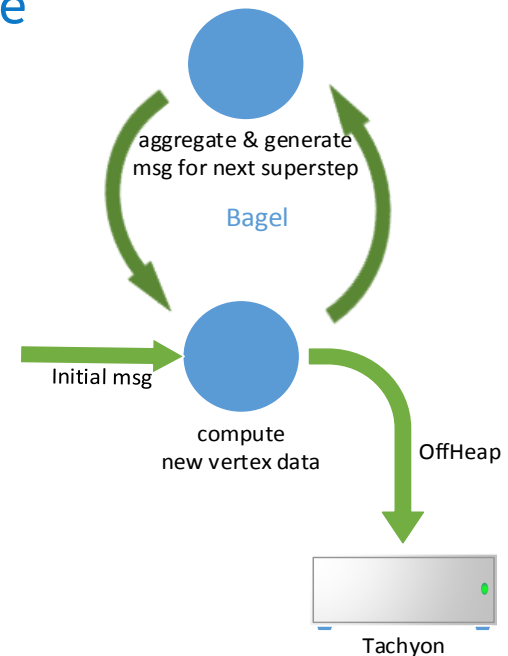
Caching data size

Iteration	Total Cached Size (before optimize)	Total Cached Size (unpersist)	Total Cached Size (with Serialization)
Initial	4.3G	4.3G	1.3G
1	12.5G	8.2G	2.9G
2	111.3G	98.8G	33.2G
3	202.1G	90.8G	30.6G

- The memory usage gets **> 50% off with unpersist in time**
- The memory usage gets another **> 60% off with serialization**

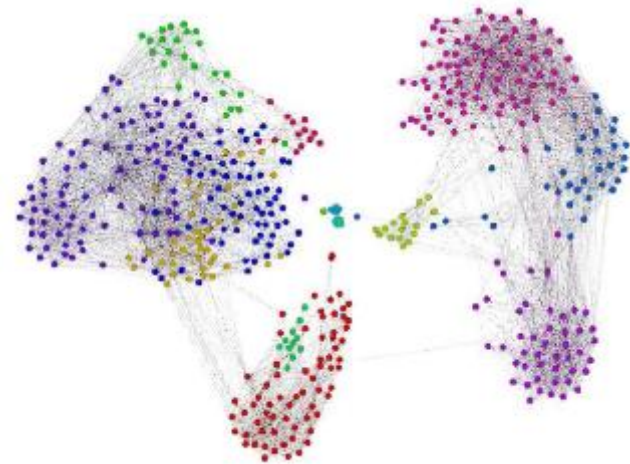
Use Tachyon to boost computation

- Huge GC overhead with large input data size
- Solution
 - Cache intermediate data into Tachyon
 - GC will not manage the data stored in Tachyon
- Optimization result
 - Brings **> 30%** performance gain by eliminate GC overhead on caching data



Graph Analysis - Clustering

- Group the videos into clusters for recommendation purposes
 - Essentially a community detection problem
 - Video groups may overlap
 - The relation (edges with weight) between videos comes from results of another recommendation engine

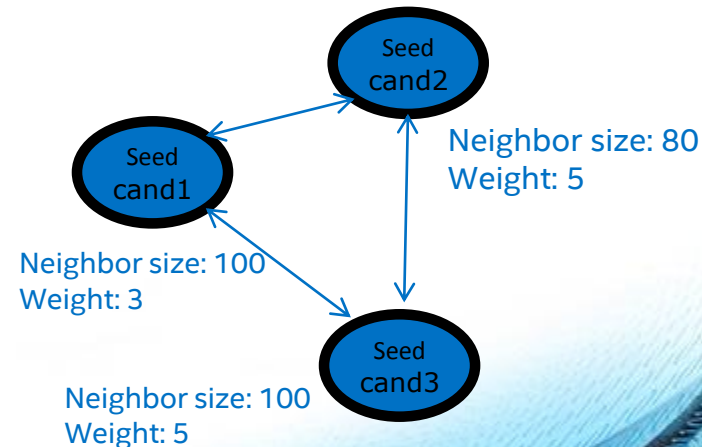


Seed Generation

- Find seeds which are the center of the clusters
 - Get top popular nodes as candidate seeds
 - Search H-hop neighbors from seed candidates
 - Solve collision during searching, remove weaker seed candidates
 - Collision means two seeds find each other, remove “weaker” one from candidate list
 - To considerations to measure “weaker”
 - The size of neighbor clustered by the seed: smaller is weaker
 - The weight of the seed: smaller is weaker

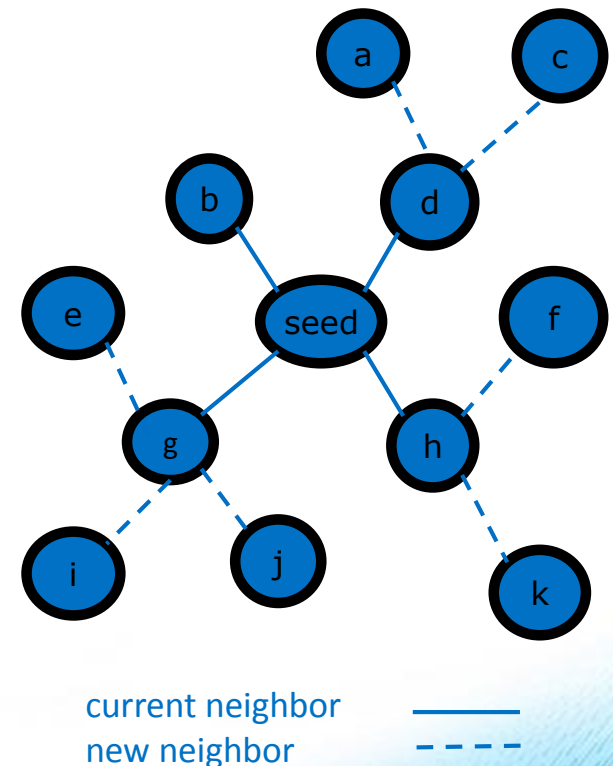
- For example: 3 seed candidates collides with each other
 - Seed1 collides with seed2, seed2 is weaker
 - Seed1 collides with seed3, seed1 is weaker
 - Seed2 collides with seed3, seed2 is weaker

After proceeding, only seed3 is left as seed candidate



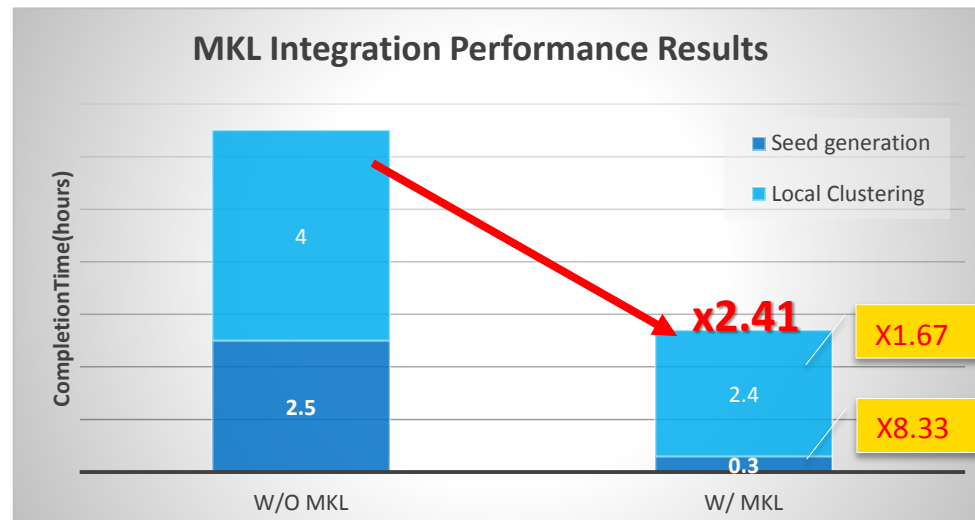
Local Clustering & Post Processing

- Local Clustering : Searching and Expanding around seeds to get clusters
 - Consider current neighbor and next new neighbor as a bipartite graph, calculate density of each sub-graph, choose the densest one and add it in to cluster
 - If cluster does not grow, mark it as not active, stop calculate on it.
 - Also need to solve the collision, the way is just like H-Hop
- Post processing: improve coverage
 - Many nodes may not be clustered
 - Calculate density between not clustered node and clusters, put the node into densest one



Speedup calculation by MKL

- Workload involves much matrix computation
 - Dominates most of the CPU time
 - Requires sparse matrix libs
- Our approach brings x2-4 speedup
 - Native math library
 - CPU instruction level optimization

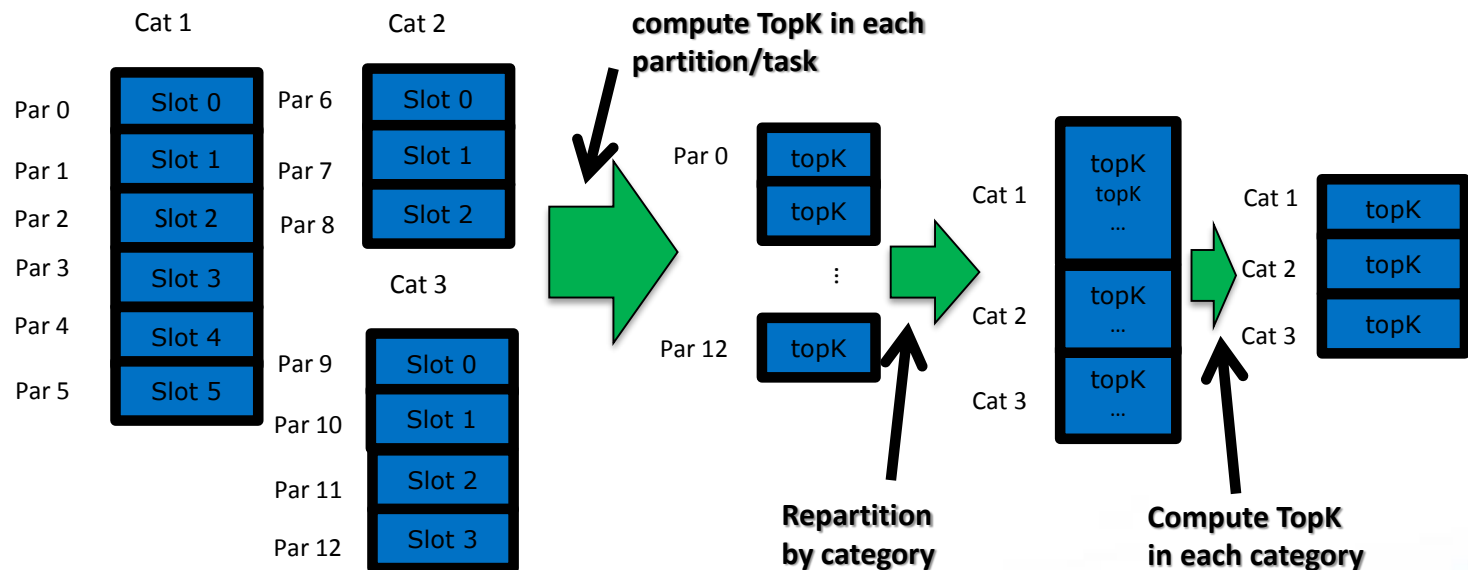


TopN in each Category

- Calculate the top-N viewed videos in each category
 - The input data:
<Category, VideoId, UserId, TimeStamp...>
 - `select category, video, sum (count) as v from testdata group by cat ,video order by v desc limit N where category='a';` (60+ categories)
 - The output data:
<Category, No, VideoId, VisitTimes>
- Data skew exists which is the key issue
 - Not balanced
 - The task processing the skewed category is much longer than others.
- Speedup from 1 hour to several minutes

Solve data skew in topN

- How to solve?
 - Divide and Conquer



Time Series Analysis: Unique Event Occurrence

- Computing unique event occurrence across the time range

- The input:

- `<TimeStamp, ObjectId, EventId, ...>`

- E.g., watch of a particular video by a specific user

- E.g., transactions of a particular stock by a specific account

- The output:

- `<ObjectId, TimeRange, Unique Event#, Unique Event(≥ 2)#, ..., Unique Event($\geq n$)#>`

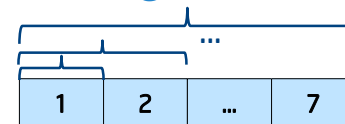
- E.g., unique event# for each day in a week (starting from Monday)

- Implementation

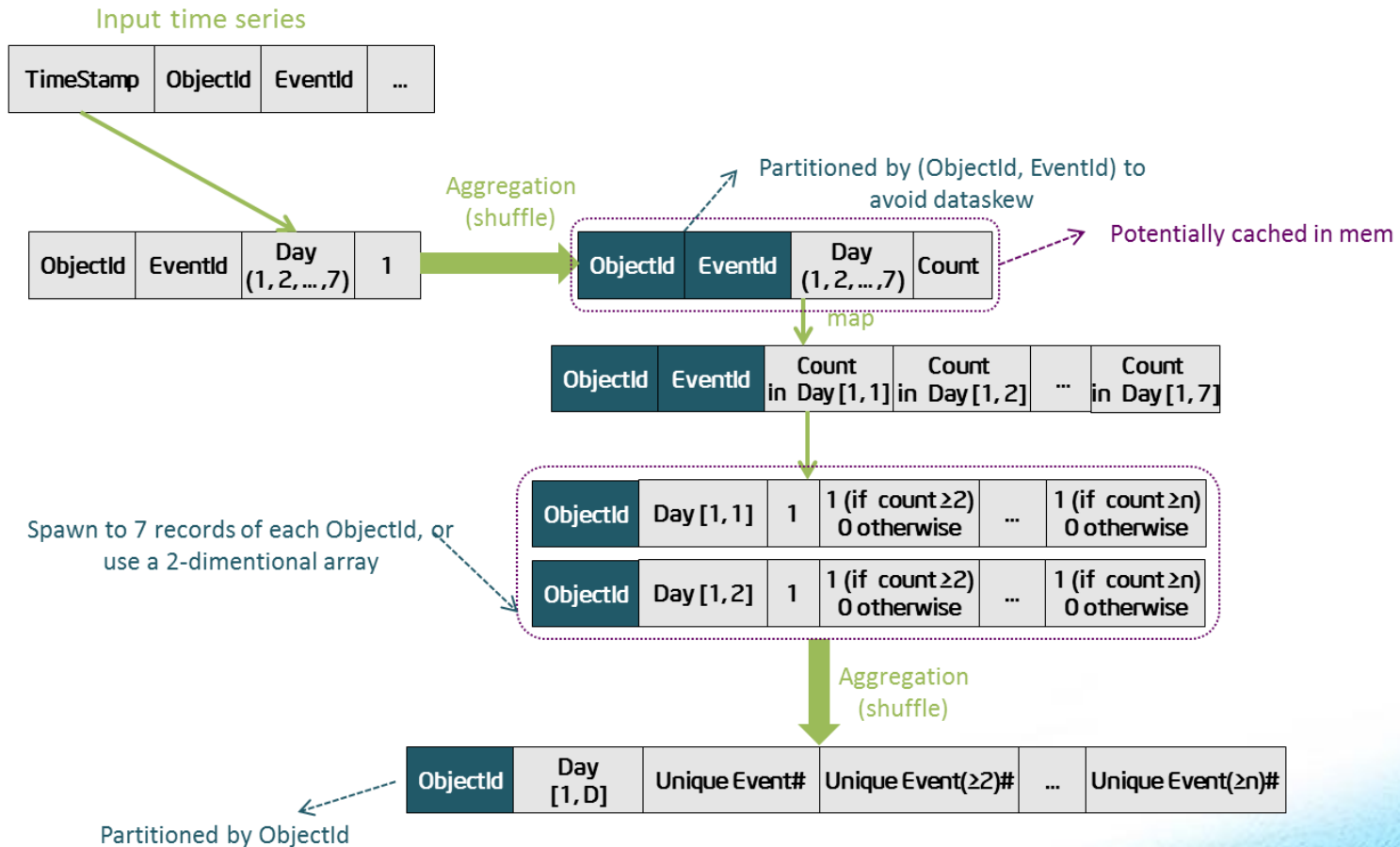
- 2-level aggregation using Spark

- Specialized partitioning, general execution graph, in-memory cached data

- Speedup from 20+ hours to several minutes

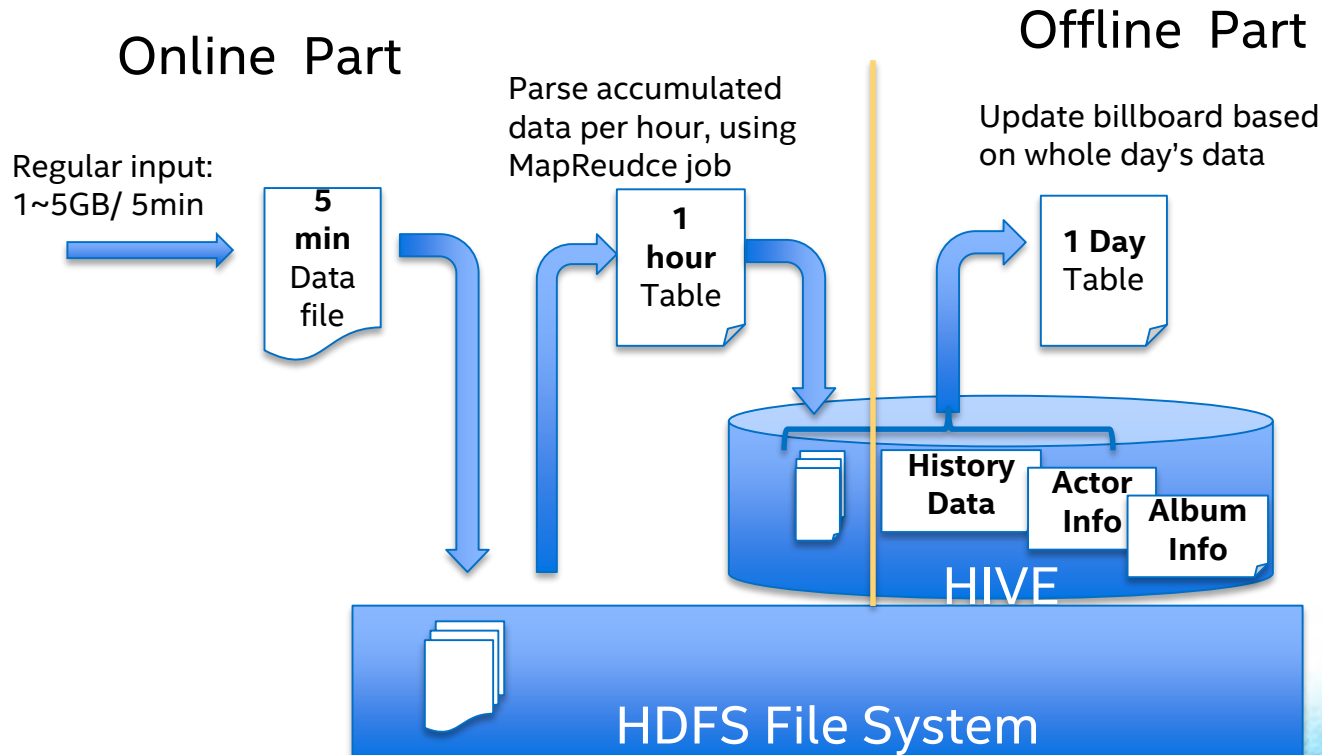


Optimization Strategy



Top N Billboard

- A full pipeline feeding data to the Top N video ranking board (updated hourly and daily)

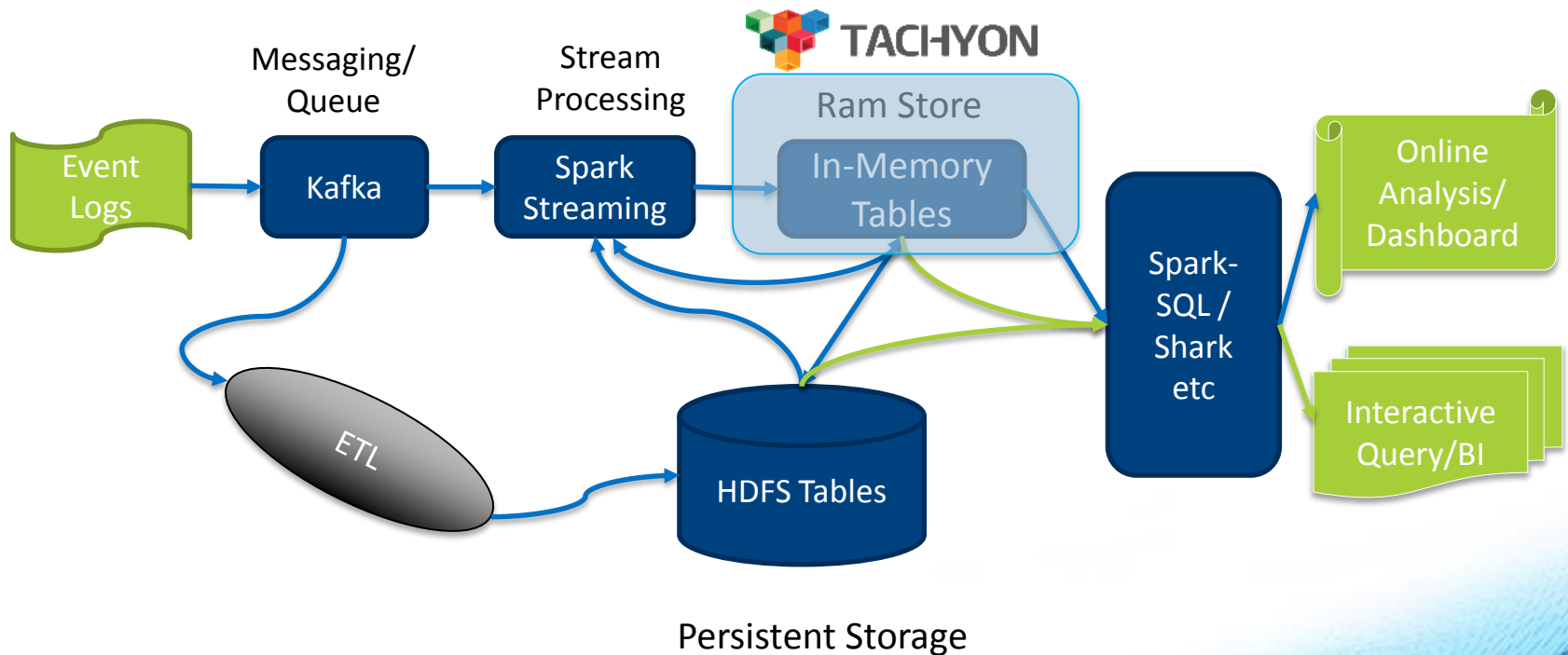


Optimization for BillBoard

- Use Shark to replace hive in the backend analysis
 - Use Spark as underlying execution engine
 - Can be easily migrated to SparkSQL
- Speeds up at least $\sim 2x$ vs. Hive
 - Avoid multiple read / write over HDFS

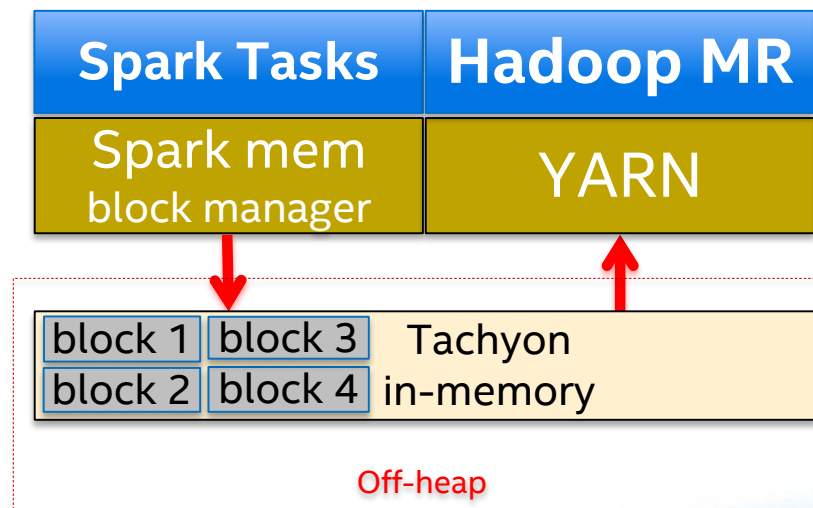
Streaming log processing

- Sharing in-memory data among different apps/frameworks



Optimize data sharing through Tachyon

- Separating the front end and back end
 - Simplify the data processing flow
 - Make the processing more stable
- Data sharing among different frameworks
 - Supports different front end and back end
- No GC overhead on shared data



In memory data sharing

Agenda

- Who are we?
- Case study and optimization experience
 - Machine Learning(Graph analysis)
 - Batch style analysis
 - Batch style OLAP
 - Streaming & Interactive OLAP
- Dew – Assistant for workload tuning on Spark
- Lessons learned & Summary

What is Dew ?

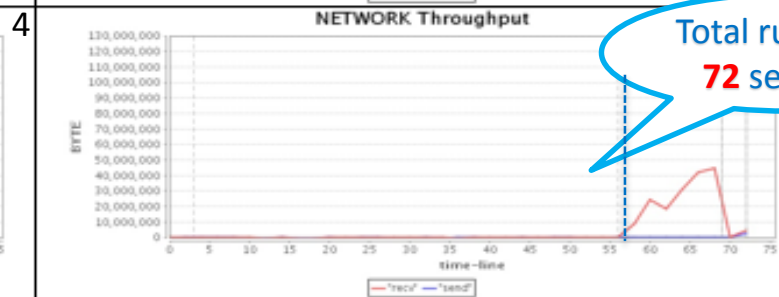
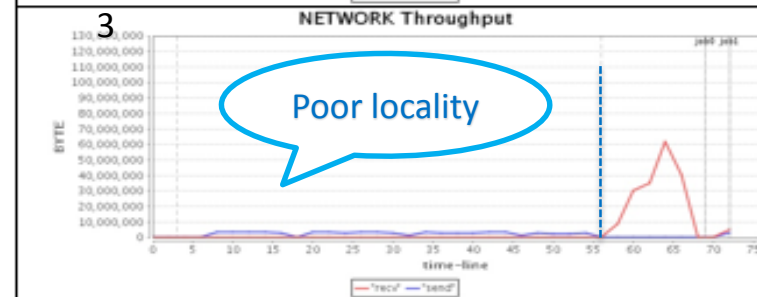
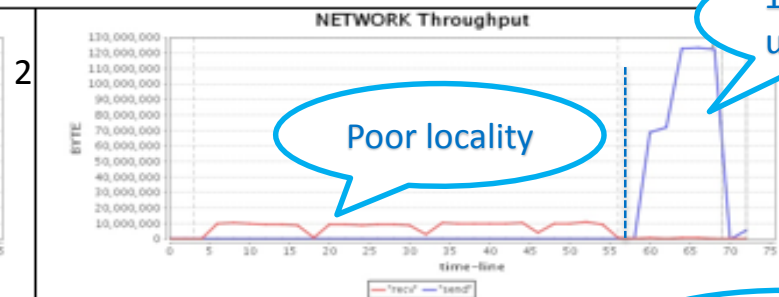
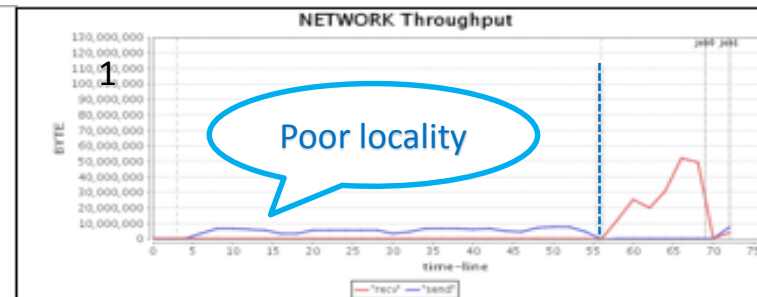
- Dew is a performance analysis tool
 - Scalable, lightweight, extensible performance analyzer
 - Dataflow-based performance analysis charts
- It will be open source soon

Tuning workload with Dew

- How to tuning Spark workload with Dew
 - System utilization monitoring
 - Task execution log analysis
- Experience with two cases
 - Task locality issue in scheduler during cold start
 - Synchronization issue during RDD cache with MEM_AND_DISK & MEM_AND_DISK_SER

Task locality issue

- Problem Statement:
 - *Extra network transportation*
 - *Network bottleneck* in immediately following stage sometimes
- Poor locality for not enough executors registered

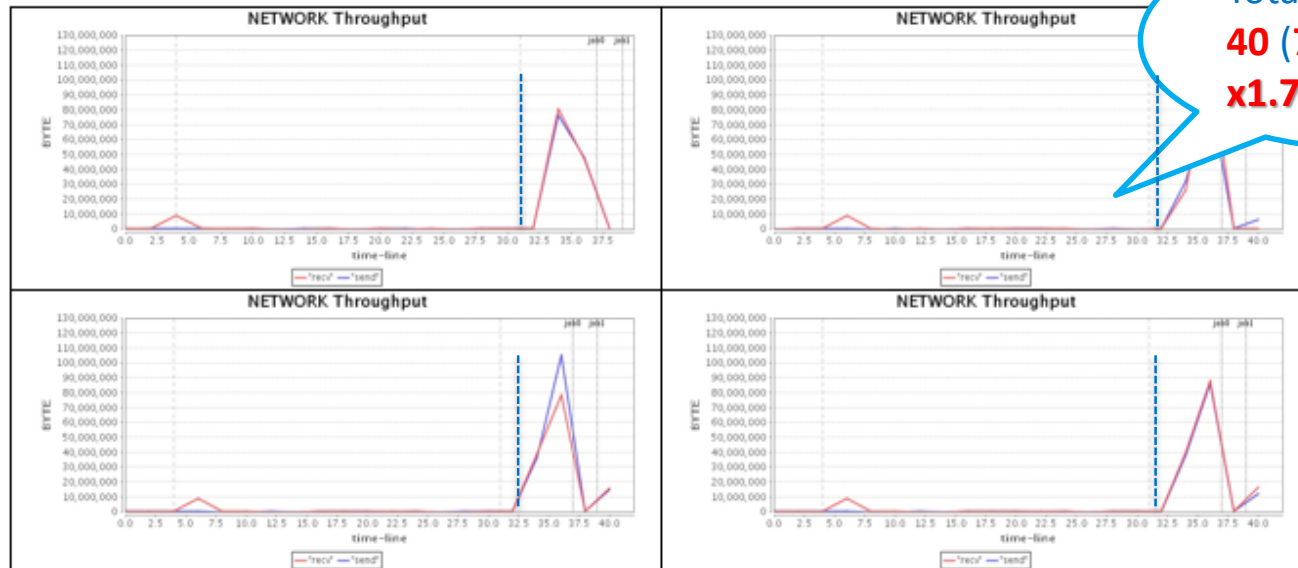


Improving task locality in scheduler

- Wait till enough executors ready [SPARK-1946](#), [SPARK-2635](#)

- Tunable knobs:

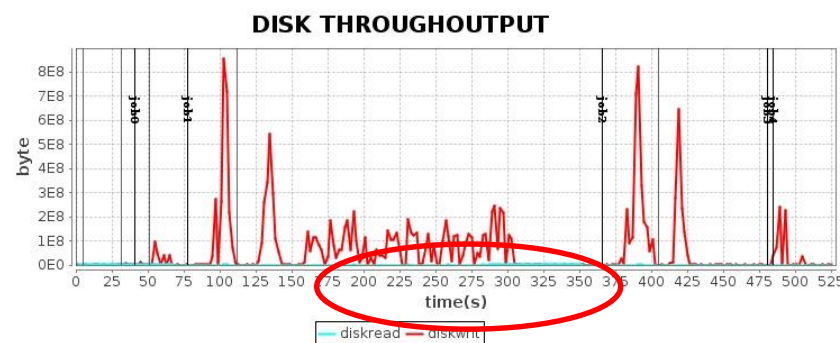
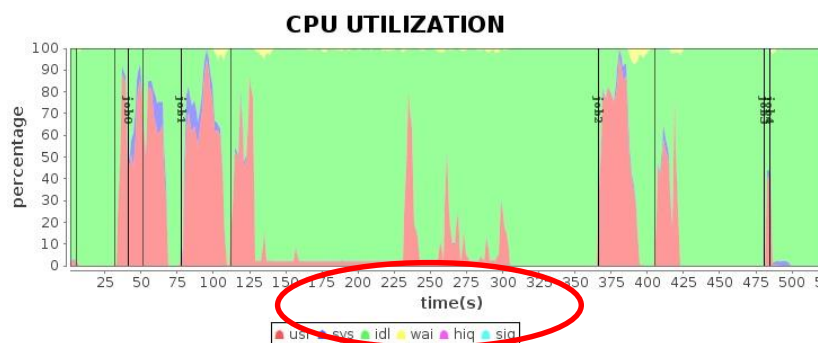
spark.scheduler.minRegisteredResourcesRatio,
spark.scheduler.maxRegisteredResourcesWaitingTime



Total run time:
40 (72) seconds,
x1.75 speedup

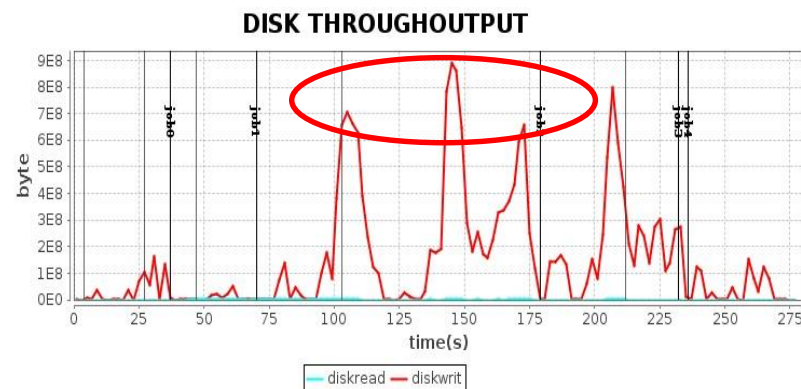
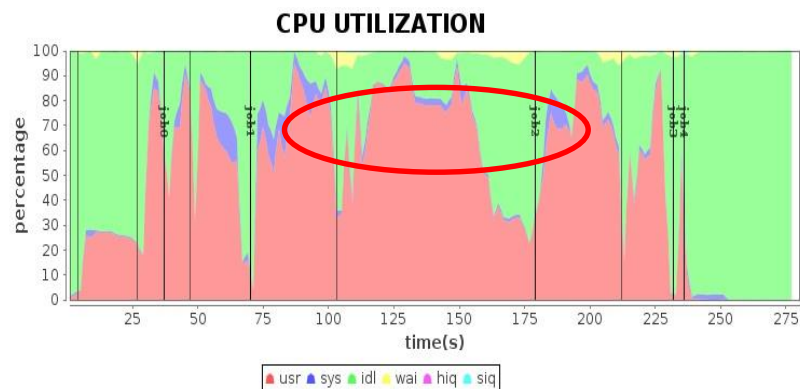
Synchronization issue in RDD Cache

- Usually large overflow data onto disk in RDD cache
 - To choose MEMORY_AND_DISK / MEM_AND_DISK_SER as preferred storage level
- Problem statement:
 - Exists synchronization issue while flushing to disks (i.e., only single HDD BW is used during flushing)
 - Comma separated storage list in Spark doesn't help



Spread out IO loads in RDD Cache

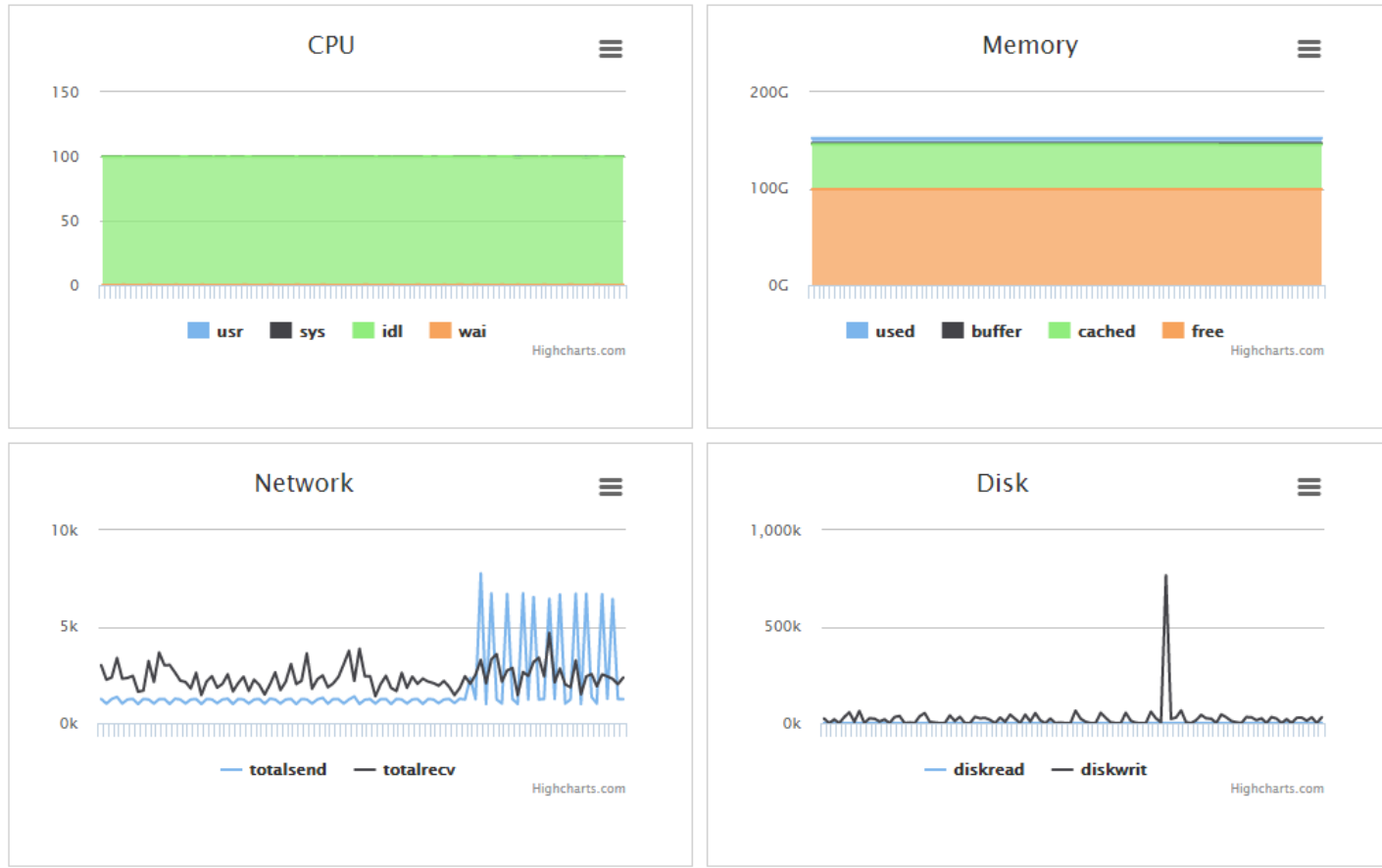
- Resolve the synchronization issue in data spill (SPARK_3000)
- Increased concurrent IO throughputs, leading to higher CPU utilization
- Speedup **x3+** in RDD cache



Also a application management system

- WebCenter: Big Data App Management
 - Cluster performance monitor
 - Application registration and execution
 - Application execution result report and analysis
- Distributed log collection and query
- Distributed command execution

Cluster Performance Status



Application & Job Registration

Add New Application

Name

kmeans

Host

sr145

Path

/home/username/workload/kmeans

Executable

./run.sh

Strategy

reExecute ▼

Type

spark ▼

Submit

Add New Job

Name

daily

Defination

nweight,wordcount

Cycle

0 0 2

Submit

Execution Result Report

Application Record List

AppName	StartTime	EndTime	Result	Operation
test1	3/5/15 12:56:00 PM.512	3/5/15 12:57:09 PM.565	success	Analysis LogQuery Diagnosis DriverLog
test1	3/4/15 12:56:00 PM.077	3/4/15 12:57:06 PM.458	success	Analysis LogQuery Diagnosis DriverLog
test1	3/3/15 12:56:00 PM.122	3/3/15 12:57:06 PM.241	success	Analysis LogQuery Diagnosis DriverLog
test1	2/11/15 11:14:18 AM.916	2/11/15 11:15:26 AM.452	success	Analysis LogQuery Diagnosis DriverLog
test1	2/11/15 9:28:59 AM.589	2/11/15 9:30:10 AM.583	success	Analysis LogQuery Diagnosis DriverLog
test1	2/6/15 3:06:55 PM.842	2/6/15 3:08:01 PM.985	failure	Analysis LogQuery Diagnosis DriverLog
test1	2/6/15 3:01:15 PM.239	2/6/15 3:02:11 PM.310	failure	Analysis LogQuery Diagnosis DriverLog

Job Record List

JobName	StartTime	EndTime	Result
app1	3/5/15 12:56:00 PM.004	3/5/15 12:56:00 PM.004	success
app1	3/4/15 12:56:00 PM.020	3/4/15 12:57:06 PM.458	success
app1	3/3/15 12:56:00 PM.042	3/3/15 12:57:06 PM.241	success
app1	2/11/15 11:14:18 AM.839	2/11/15 11:15:26 AM.452	success
app1	2/11/15 9:28:59 AM.513	2/11/15 9:30:10 AM.583	success
app1	2/6/15 3:06:55 PM.724	2/6/15 3:08:01 PM.985	failure
app1	2/6/15 3:01:15 PM.209	2/6/15 3:02:11 PM.310	failure
app1	2/6/15 2:58:23 PM.768	2/6/15 2:59:19 PM.838	failure
app1	2/6/15 2:55:13 PM.727	2/6/15 2:56:19 PM.657	success

Diagnosis

Show DiagnosisResult

hostName	diagnosisName	level	describe	advice
sr453	waste-CPU	middle	Cpu resources waste percent is 69.76%. More time on non-computation task.	Improve node's disk and network performance.
sr454	waste-CPU	middle	Cpu resources waste percent is 69.09%. More time on non-computation task.	Improve node's disk and network performance.
sr453	load-Net-Send	high	load-Net-Send is lower than cluster average by 81.74%	Check the node or your application algorithm.

Log Collection

Contents of directory [/dewlog/application_1422431846398_0127](#)

Goto : go

[Go to parent directory](#)

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
driver.log	file	191.54 KB	3	128 MB	2015-03-05 14:11	rw-r--r--	liyezhan	supergroup
sr453.container_1422431846398_0127_01_000002.stderr	file	15.37 KB	3	128 MB	2015-03-05 14:11	rw-r--r--	liyezhan	supergroup
sr453.container_1422431846398_0127_01_000002.stdout	file	0 B	3	128 MB	2015-03-05 14:11	rw-r--r--	liyezhan	supergroup
sr453.container_1422431846398_0127_01_000004.stderr	file	14.86 KB	3	128 MB	2015-03-05 14:11	rw-r--r--	liyezhan	supergroup
sr453.container_1422431846398_0127_01_000004.stdout	file	0 B	3	128 MB	2015-03-05 14:11	rw-r--r--	liyezhan	supergroup
sr453.container_1422431846398_0127_01_000006.stderr	file	16.80 KB	3	128 MB	2015-03-05 14:11	rw-r--r--	liyezhan	supergroup
sr453.container_1422431846398_0127_01_000006.stdout	file	0 B	3	128 MB	2015-03-05 14:11	rw-r--r--	liyezhan	supergroup
sr453.container_1422431846398_0127_01_000008.stderr	file	18.38 KB	3	128 MB	2015-03-05 14:11	rw-r--r--	liyezhan	supergroup
sr453.container_1422431846398_0127_01_000008.stdout	file	0 B	3	128 MB	2015-03-05 14:11	rw-r--r--	liyezhan	supergroup

Log Query

Query Result

```
sr454.container_1422431846398_0183_01_000001.stderr 15/04/01 12:56:16 INFO yarn.ExecutorRunnable: Setting up executor with
commands: List($JAVA_HOME/bin/java, -server, -XX:OnOutOfMemoryError='kill %p', -Xms4096m, -Xmx4096m, -Djava.io.tmpdir=$PWD/tmp,
'-Dspark.driver.port=38496', '-Dspark.akka.timeout=600', '-Dspark.akka.frameSize=1000', -Dspark.yarn.app.container.log.dir=<LOG_DIR>,
org.apache.spark.executor.CoarseGrainedExecutorBackend, --driver-url, akka.tcp://sparkDriver@sr145:38496
/user/CoarseGrainedScheduler, --executor-id, 8, --hostname, sr453, --cores, 1, --app-id, application_1422431846398_0183, --user-
class-path, file:$PWD/_app_.jar, 1>, <LOG_DIR>/stdout, 2>, <LOG_DIR>/stderr)
sr454.container_1422431846398_0183_01_000001.stderr 15/04/01 12:56:16 INFO yarn.ExecutorRunnable: Setting up executor with
commands: List($JAVA_HOME/bin/java, -server, -XX:OnOutOfMemoryError='kill %p', -Xms4096m, -Xmx4096m, -Djava.io.tmpdir=$PWD/tmp,
'-Dspark.driver.port=38496', '-Dspark.akka.timeout=600', '-Dspark.akka.frameSize=1000', -Dspark.yarn.app.container.log.dir=<LOG_DIR>,
org.apache.spark.executor.CoarseGrainedExecutorBackend, --driver-url, akka.tcp://sparkDriver@sr145:38496
/user/CoarseGrainedScheduler, --executor-id, 10, --hostname, sr453, --cores, 1, --app-id, application_1422431846398_0183, --user-
class-path, file:$PWD/_app_.jar, 1>, <LOG_DIR>/stdout, 2>, <LOG_DIR>/stderr)
sr454.container_1422431846398_0183_01_000001.stderr 15/04/01 12:56:16 INFO yarn.ExecutorRunnable: Setting up executor with
commands: List($JAVA_HOME/bin/java, -server, -XX:OnOutOfMemoryError='kill %p', -Xms4096m, -Xmx4096m, -Djava.io.tmpdir=$PWD/tmp,
'-Dspark.driver.port=38496', '-Dspark.akka.timeout=600', '-Dspark.akka.frameSize=1000', -Dspark.yarn.app.container.log.dir=<LOG_DIR>,
org.apache.spark.executor.CoarseGrainedExecutorBackend, --driver-url, akka.tcp://sparkDriver@sr145:38496
/user/CoarseGrainedScheduler, --executor-id, 9, --hostname, sr454, --cores, 1, --app-id, application_1422431846398_0183, --user-
class-path, file:$PWD/_app_.jar, 1>, <LOG_DIR>/stdout, 2>, <LOG_DIR>/stderr)
sr454.container_1422431846398_0183_01_000001.stderr 15/04/01 12:56:16 INFO yarn.ExecutorRunnable: Setting up executor with
commands: List($JAVA_HOME/bin/java, -server, -XX:OnOutOfMemoryError='kill %p', -Xms4096m, -Xmx4096m, -Djava.io.tmpdir=$PWD/tmp,
'-Dspark.driver.port=38496', '-Dspark.akka.timeout=600', '-Dspark.akka.frameSize=1000', -Dspark.yarn.app.container.log.dir=<LOG DIR>,
```

Agenda

- Who are we?
- Case study and optimization experience
 - Machine Learning(Graph analysis)
 - Batch style analysis
 - Batch style OLAP
 - Streaming & Interactive OLAP
- Dew – Assistant for workload tuning on Spark
- Lessons learned & summary

Lessons we learned

- Better memory management
 - More efficient use of on heap memory
 - Reclaim memory in time & data serialization
 - Manage data using off heap storage
 - GC elimination & in memory data sharing
- balance the workload
 - Solve data skew
 - Spread the system load
 - Better task / data locality
- Improve the computation
 - JNI lib for performance sensitive code
 - High performance library

Summary

- Spark plays an important role in big data
- Continuously mature Spark for next-gen big data on IA altogether
- Areas we focus on:

Performance & scalability

- Public Regression test report
- Pluggable storage in Spark
- Aggregation optimization in Spark-SQL
- Tachyon hierarchical storage

Reliability, Usability

- StreamSQL
- Spark-streaming HA
- SparkR

Notices and Disclaimers

- No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.
- Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.
- This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.
- The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.
- Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.
- Intel, the Intel logo, {List the Intel trademarks in your document} are trademarks of Intel Corporation in the U.S. and/or other countries.
- *Other names and brands may be claimed as the property of others
- © 2015 Intel Corporation.

Disclaimers - Continued

- Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.
- For more information go to <http://www.intel.com/performance>.