Experience and Lesson Learned in Building Real-World Spark Applications

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

This is team work

Thanks all the great guys' contributions in last 6 months!

Spark is Sparkling in China

- Spark is not only rising but also sparkling
  - Evolving from promotion to demand growth
  -  +  co-locates there
- Intel partnering with several large web sites in China since 2012
  - Building real-world big data analytic applications using Spark stack
  - E.g., Alibaba, Baidu iQiyi, Youku, Sina, etc.
- Application focuses
  - Iterative, complex machine learning & graph analysis
  - Real-time analytical processing (RTAP)
  - Complex, Interactive OLAP/BI

http://spark-summit.org/wp-content/uploads/2013/10/Spark_Summit_2013_Jason_Dai.pdf

Lessons learned

1. Manage Memory
2. Avoid Network
3. Improve Disk I/O
4. Optimize Computations

Lessons learned

1. Manage Memory
2. Avoid Network
3. Improve Disk I/O
4. Optimize Computations

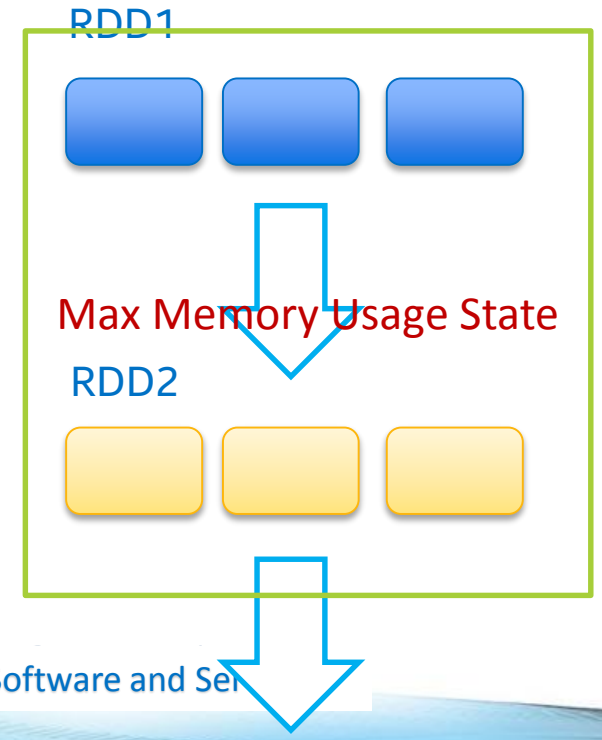
Manage Memory

Memory

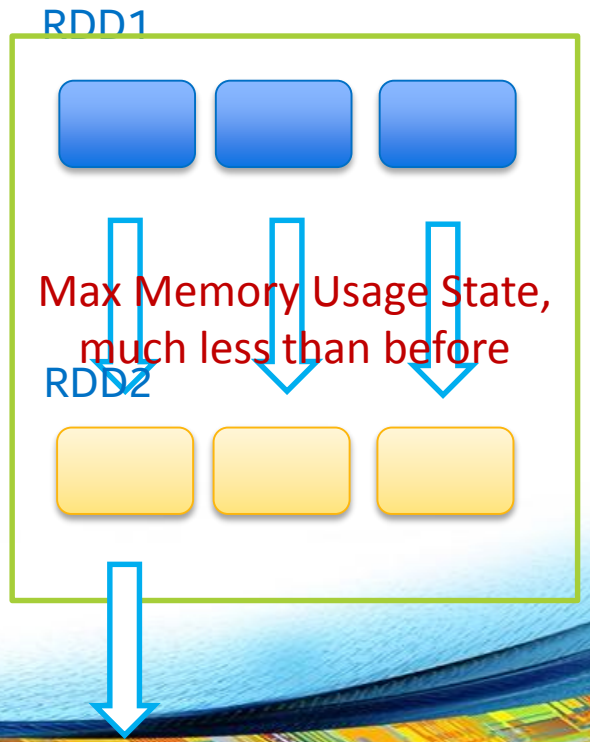
- Graph analysis – Ndegree association
  - Computing associations between two vertices that are n-hop away (E.g., Friends of friend)
- Graph-parallel implementation by Bagel or GraphX
  - High memory utilization for cached messages in Bagel
- Memory space not used efficiently & huge GC impact
  - Memory is not freed timely
  - Vertex duplication for sparse graph in GraphX

Solution#1 Partition based RDD un-persistent

- RDD based un-persistent



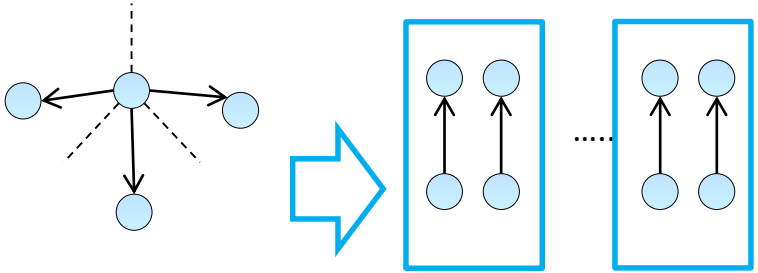
- Partition based un-persistent



Solution#2 Reduce vertex duplication

Memory

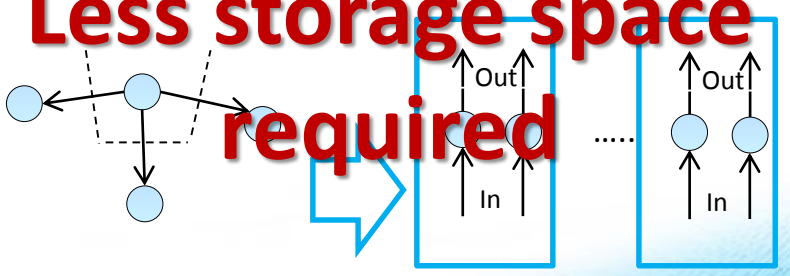
- Vertex cut
  - Replicated cross different partitions
  - Vertex data is immutable and updated from time to time



Vertex cut Partition by Edges

- Edge cut
  - Replicated cross different partition
  - Edge data is invariable and re-usable
  - Sparse graph leads to weak connectivity and small edge number
  - Single end-point based computation*

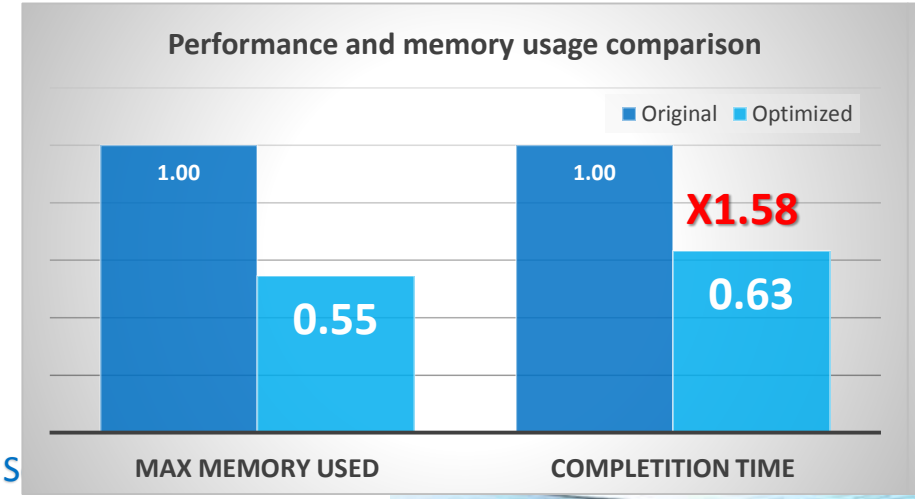
Less storage space required

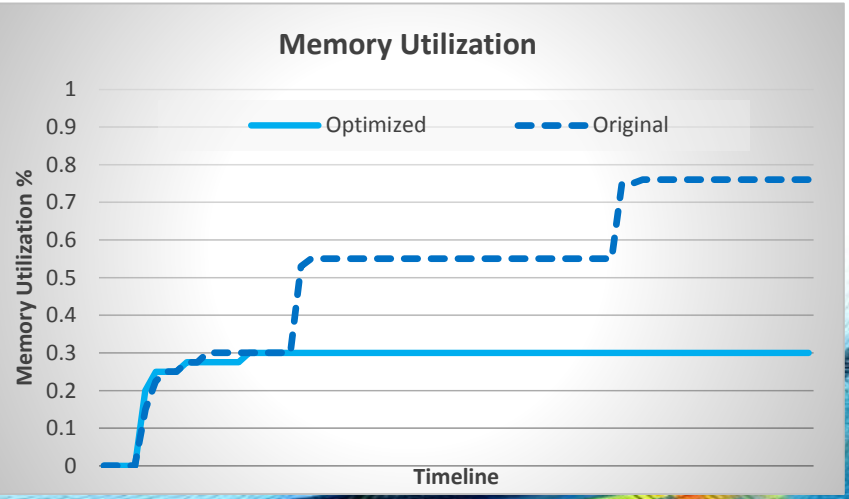


Edge cut Partition by Vertex

Manage Memory

- More efficient memory space usage leads to better performance(**x1.58**)
  - Support “Edge cut” partition policy
  - Partition based RDD un-persistent mechanism (smaller memoryFraction)





Manage Memory

Memory

- Top N video ranking case
  - Video ranking board generated periodically (hourly and daily)
- Using Shark to implement chaining ETL and complex analytical queries
- Join is frequently used, and its optimization matters a lot
  - E.g., multi-way join
- Hard to fit entire table in memory
- OFFHEAP(Tachyon) cache leads better performance

Lessons learned

1. Manage Memory
2. Avoid Network
3. Improve Disk I/O
4. Optimize Computations

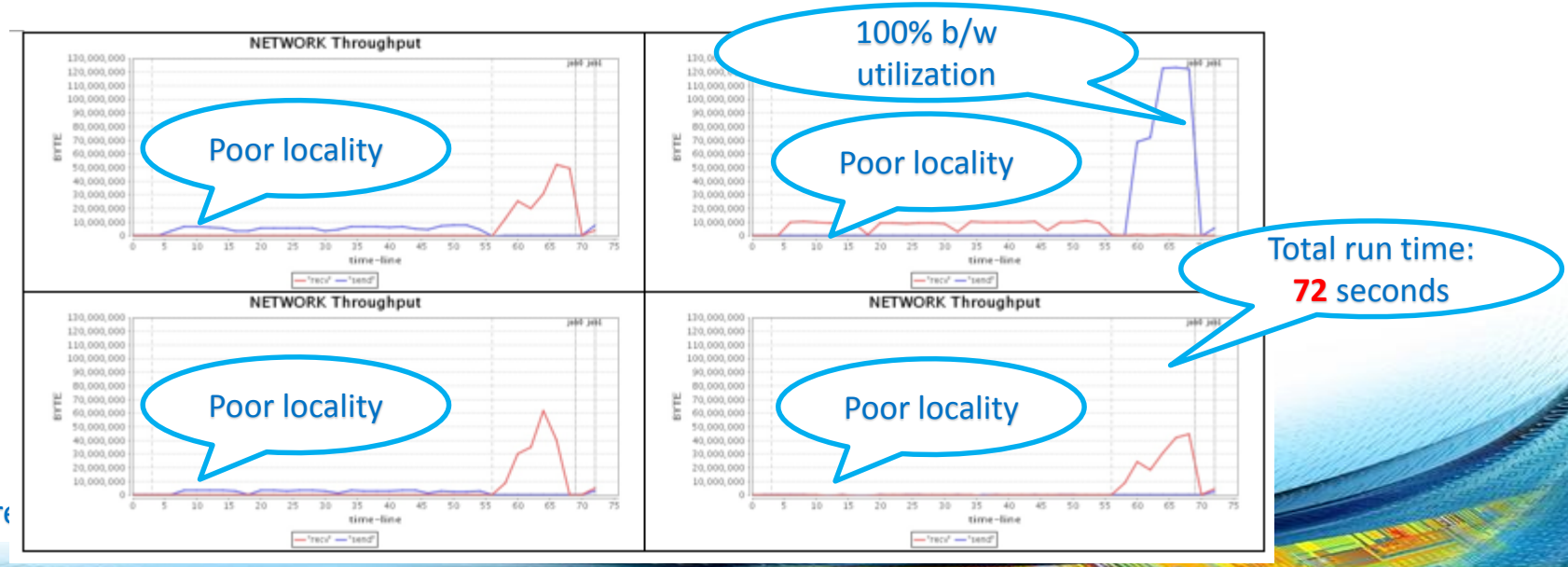
Avoid Network

Network

- Graph analysis – Ndegree association
  - Computing associations between two vertices that are n-hop away
- Poor data locality (assign to pendingTasksWithNoPrefs)
  - Introduce extra network traffic
  - single node network bottleneck
- Entitle NODE_LOCAL tasks to ANY needs certain delay
  - By default 3 seconds

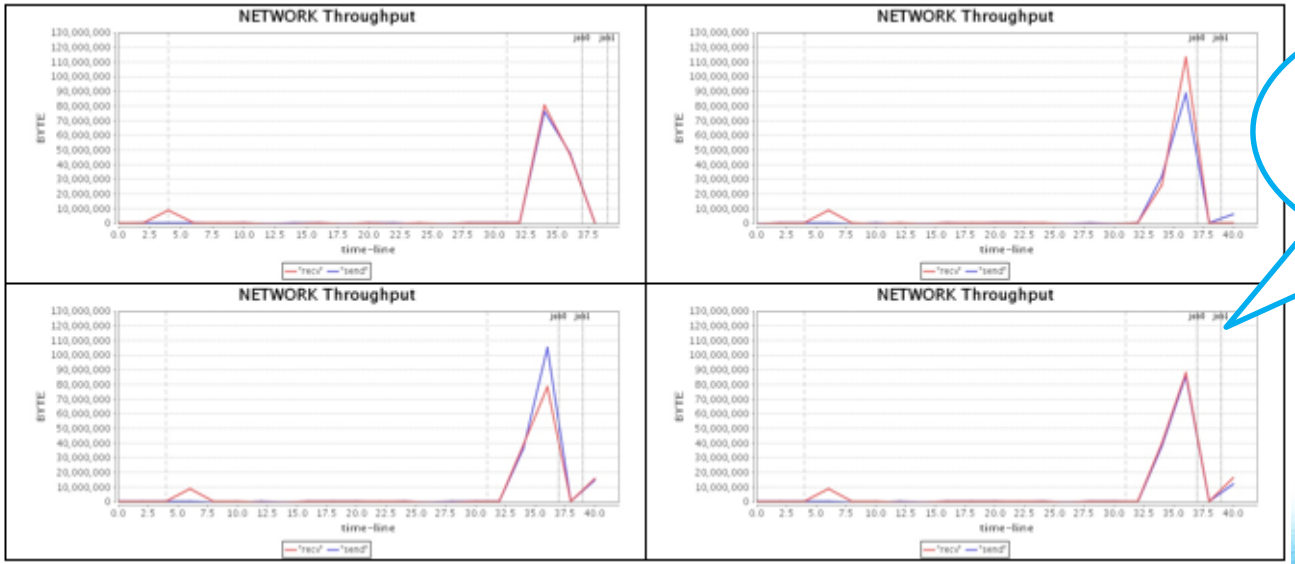
Improving task locality in scheduler(before)

- Wait till enough executors registered SPARK-1946
- Prioritized locality list SPARK-2193



Improving task locality in scheduler(after)

- Wait till enough executors registered SPARK-1946
- Prioritized locality list SPARK-2193



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

Network

Lessons learned

1. Manage Memory
2. Avoid Network
3. Improve Disk I/O
4. Optimize Computations

Improve Disk I/O

- Top N video ranking case
  - Video ranking board generated periodically (hourly and daily)
- Multiple stages in one query share single mapred.reduce.tasks number
- It's hard to set a proper "mapred.reduce.tasks" manually.
  - Too small reduce task number causes Out Of Memory
  - Too large reduce task number causes performance degradation

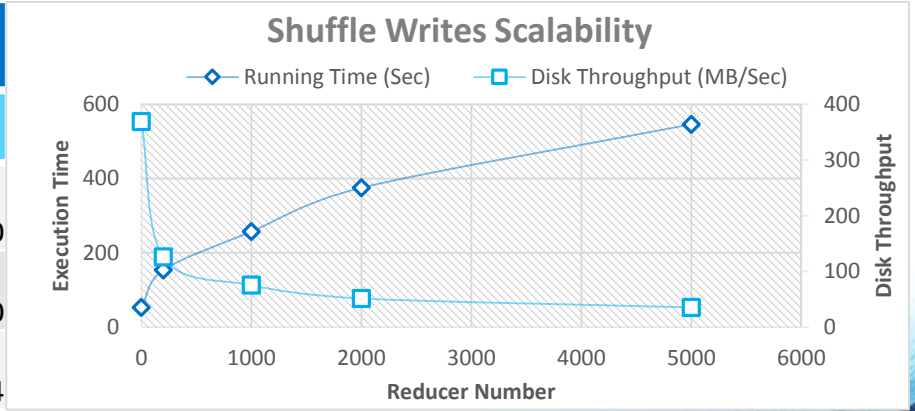
Disk I/O

#B
#D

Random IO access ruins the Shuffle writes

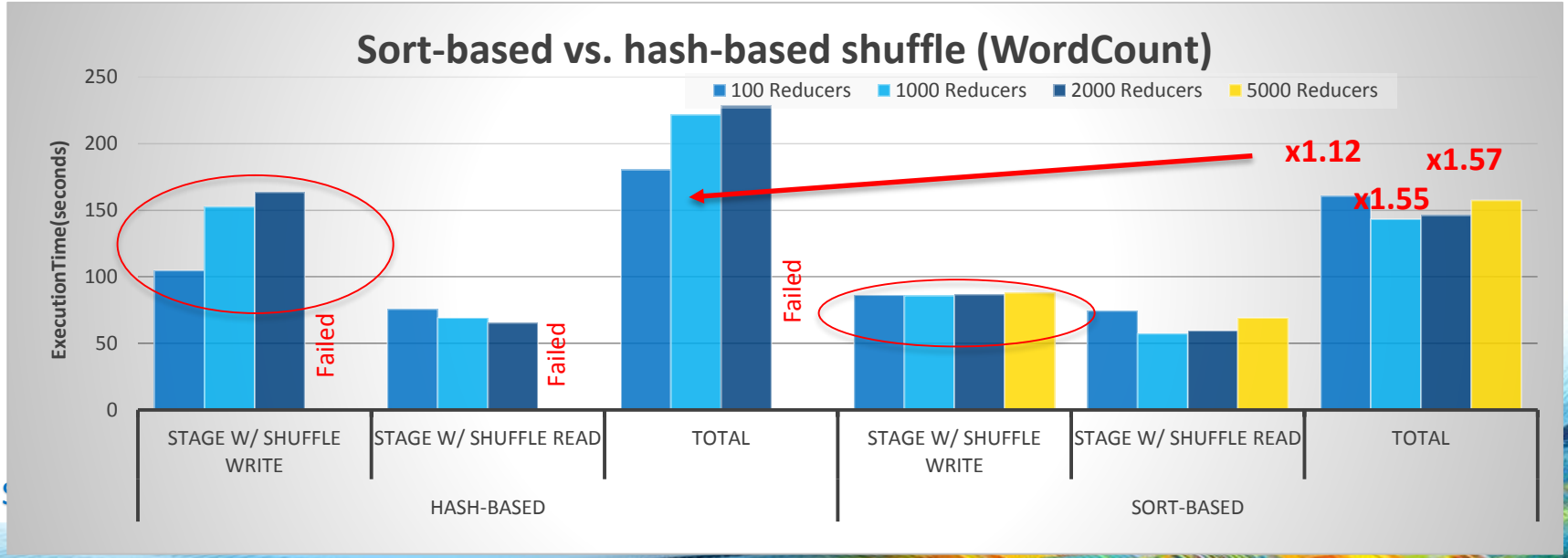
- Increasing Reducer# causes more outstanding IO requests, lower disk I/O BW.
- Further consolidation is required for higher scalability
  - E.g., sort-based shuffle in Hadoop MR

Normalized completion time with different Reducer#							
Reducer Number	Q8	Q9	Q10	Q11	Q12	Q13	Q14
96	1.00	1.00	1.00	1.00	1.00	1.00	1.00
384	3.20	1.67	1.72	1.94	1.00	1.63	2.00
2048	44.27	6.28	6.72	7.56	1.00	5.88	9.94



Sort-based Shuffle (POC)

- Consolidate to 1 output file in shuffle write
- Sort-based shuffle performs better than Hash-based(**~x1.12-1.57**)
- More scalable shuffle write



Lessons learned

1. Manage Memory
2. Avoid Network
3. Improve Disk I/O
4. Optimize Computations

Optimize Computations

- Top N video ranking case
  - Video ranking board generated periodically(hourly and daily)
- Kinds of complex and repeated expressions In the real-world cases
- Requires expression optimization in Shark
- Shark CodeGen brings **x1.2** speedup for the real world case
  (https://github.com/amplab/shark/tree/branch-0.9-codegen)

```
…
 select if(t1.xxx is null, t1.pid, t2.pid) as n_pid,
        if(t1.xxx is null, t1.cid, t2.cid) as n_cid,
        if(t1.xxx is null, t1.aid, t2.aid) as n_aid,
        if(t1.xxx is null, t1.tid, t2.tid) as n_tid,
        if(t1.xxx is null, t1.yyy, t1.xxx) as n_vid,
        if(t1.xxx is null, 0, t2.sum_vv) as col_vv,
        if(t1.yyy is null, 0, t1.col_vv_his) as
col_vv_his_pre,
        if(t1.xxx is null, 0, t2.sum_pt) as col_pt
  from …
```

CPU

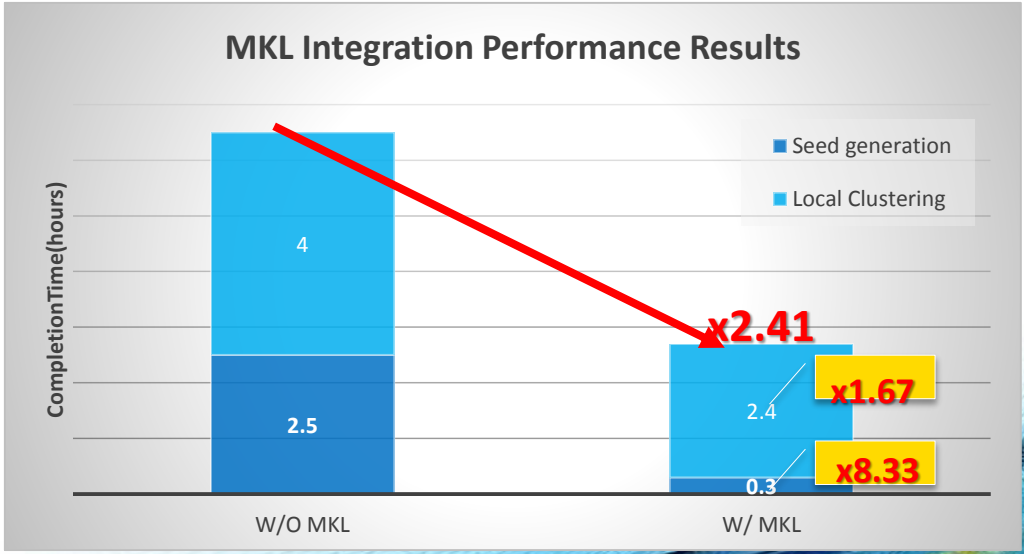
Optimize Computations

- Graph clustering case - Group the videos into clusters for recommendation
  - Essentially a community detection problem

- Need sparse matrix libs
- Complex matrix operations (e.g., multiplication) dominates CPU time

- Native math lib & CPU instruction level optimization speedup a lot

CPU



Current & Future focuses in Intel

- Plays key contributions to grow Spark and its ecosystem

- Usability & Operability
  - Metrics & monitors in Spark-streaming
  - CLI supports in Spark-sql
  - More advanced functions in Spark-sql
  - Better Tachyon integration & hierarchical store

- Performance
  - New pluggable shuffle (with Cloudera) & pluggable storage
  - Improve scheduler for better locality
  - Join &Aggregation optimization in Spark-sql

Summary

- Spark plays an important role in big data
- Less
- Con

Call for Action:
To develop workloads and do optimization together

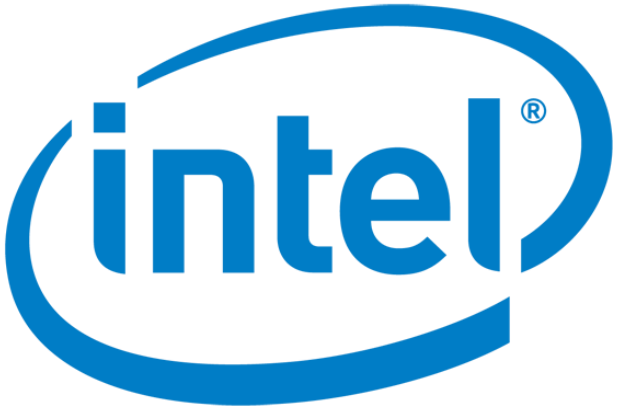
Notices and Disclaimers

- INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.
  A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.
- Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.
- The products described in this document 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.
- Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.
- Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm
- Intel, the Intel logo, Intel Xeon, and Xeon logos 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.

Disclaimers - Continued

- Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families: Go to: Learn About Intel® Processor Numbers http://www.intel.com/products/processor_number
- All the performance data are collected from our internal testing. Some results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.


intel

Configurations

	Testcase	CPU	Nodes	CPU Cores	Threads per Core	Sockets	Memory	Disks	Network
Hardware	#A	Intel Xeon WSM L5640 @ 2.27G	1+4	6	2	2	48GB	1TB * 4	1G
	#B	Intel Xeon L5640 @ 2.27 G	1+4	6	2	2	96GB	1TB * 11	1G
	#C	Intel Xeon E5-2660 @ 2.2G	1+4	8	2	2	192GB	1TB * 4	1G
	#D	Intel Xeon WSM L5640 @ 2.27G	1+4	6	2	2	48GB	1TB * 12	1G

		System	Kernel	File System	Spark stack version	Hadoop version	JDK version	Scala version
Software	#A	Ubuntu11.04	2.6.38-8-server.X86_64	Ext4	0.8.0-SNAPSHOT (Spark); internal dev branch (GraphX)	1.0.4	1.7.0_55-b13	2.9.3
	#B	RHEL6.0(Santiago)	2.6.32-71.el6.x86_64	Ext4	0.9.1(Spark); 0.9.1 Snapshort(Shark); 0.5.0-SNAPSHOT(Tachyon)	CDH 4.3.0	1.7.0_04-b20	2.10.3
	#C	RHEL6.2	2.6.32-220.el6.x86_64	Ext4	1.0.0 (Spark)	1.0.4	1.7.0_55-b13	2.10.4
	#D	Ubuntu12.04	2.6.38-8-server.X86_64	Ext4	1.1.0-SNAPSHOT	1.0.4	1.7.0_04	2.10.4