Pablo Basanta-Val. *An Efficient Industrial Big-data Engine*. IEEE Transactions on Industrial Informatics (September 2017). In press.
DOI: 10.1109/TII.2017.2755398

An Efficient Industrial Big-data Engine

Pablo Basanta-Val
pbasanta@it.uc3m.es

***Abstract*—Current trends in industrial systems opt for the use of different big-data engines as a mean to process huge amounts of data that cannot be processed with an ordinary infrastructure. The number of issues an industrial infrastructure has to face is large and includes challenges such as the definition of different efficient architecture setups for different applications, and the definition of specific models for industrial analytics. In this context, the article explores the development of a medium size big-data engine (i.e. implementation) able to improve performance in map-reduce computing by splitting the analytic into different segments that may be processed by the engine in parallel using a hierarchical model. This type of facility reduces end-to-end computation time for all segments with their results then merged with other information from other segments after their processing in parallel. This type of setup increases performance of current clusters improving I/O operations remarkably as empirical results revealed.**

***Index Terms*— industrial big-data, efficient infrastructure, map-reduce, industrial infrastructure, big-data engine**

I. Introduction

Currently there is a trend in industry 4.0 pointing to the use of big-data as a relevant element in the development of next generation industrial-systems [1][2][3][37]. Big-data offers many opportunities to evaluate data for example to identify preferences from end-users, to target fraud and also to relate other issues derived from a combined and statistical processing of data [3][5][33]. Industrial big-data refers to large amounts of time series data generated at high-speed by industrial equipment, typically located in worldwide factories [5][6][7][8][9]. Industrial big-data is used to help take decisions from basic information, so that business will be able to reduce maintenance costs offering improved services. Both, "general purpose" and "industrial big-data" share a number of defining characteristics such as volume, variety, velocity, variability, and veracity. However, industrial big-data applications add two additional V's: *visibility*, which refers to discovery of unexpected insights of existing processed data; and *value*, which puts emphasis on the objective of analytics, creating new value from data [10][11][40-44].

Another difference among traditional big-data and industrial big-data is that industrial big-data is more structured, correlated, and ready for analytics than its general purpose big-data counterpart [1][12]. This is so because industrial big-data is generated by automated equipment, where the environment and processes are more controlled than in human interactions typical of social networks. Besides 5 Vs of a big-data system, industrial big-data is characterized with new 3Bs [10][11][39]: *below-surface*, *broken*, and *bad-quality*. *Below-surface* refers to the idea of mining relationships and capturing the phenomena that are typical of an industrial system. *Broken* refers to the idea of completeness of data over its volume that it also present in many industrial systems. *Bad-quality* refers to the problem of dealing with low-quality data, which may lead to a disaster in the industrial ecosystem.

The list of technologies [14] ready to be used as a part of industrial big-data infrastructure includes key technologies such as Hadoop [15], Storm [16], and Spark [17], and other technologies that are arranged as a part of the big-data infrastructure. Complex applications require the use of online and offline technologies to meet different application requirements. Most technologies are in a consolidation process that it is going to last for next years, specially for the industrial big-data arena. Currently, there is a trend that looks for efficiency in map-reduce interactions, showing an interest in having a more efficient infrastructure [14][19][20][21][39]. The trend is of special interest in industrial systems that have to produce shorter response times to external events even as the system performance increases.

To improve the performance of current infrastructures, the article describes an optimization, called hierarchical map-reduce, under the context of an industrial infrastructure, called IBIDAE (Industrial BIg-DAta Engine), which runs on a hierarchically modified Spark and Hadoop engine. Engine refers to a motor and may be used as synonym implementation of an *architecture*. Conversely, an architecture refers to the art of designing complex systems efficiently that support applications. Lastly, applications are small programs designed to run on a specific engine of an architecture.

The main contribution of IBIDAE is a hierarchical map-reduce architecture that shows how for an industrial trace performance may be increased by running several parts of the analytic in parallel. Current engines for MapReduce and Spark do not offer a proper support to this type of facilities as a part of their distributions but they can be extended to offer this support and thus experience remarkable increases in performance.

Regarding those works related to the performance of map-reduce, the most relevant is [20] which offers four times improvements in total response-time by parallelizing reduce phases, in a map-reduce engine. The main difference among the previous approach and the proposed one is that it

Pablo Basanta Val (pbasanta@it.uc3m.es) Web Technologies Laboratory, Departamento de Ingeniería Telemática, Spain Departamento de Ingeniería Telemática and UC3M-BS, Institute of Financial Big Data, Universidad Carlos III de Madrid, Spain, 28911.

parallelizes different analytics using a double hierarchy to offer reduced response times. In addition, previous work does not create additional map-reduce tasks to be executed more efficiently in parallel. The results included in IBIDAE are relevant for architects of other infrastructures to further improve their own systems.

The rest of the article describes related work, the architecture of IBIDAE, and the performance results one may expect from this type of infrastructure and its hierarchical optimization. Section II deals with different technologies similar in goals to the IBIDAE architecture, exploring their mutual relationships. Then Section III describes the IBIDAE architecture covering architectural issues, computational model, and impact on current computational models. The empirical benefits offered by the architecture are explained later in Section IV. Section V draws conclusions and highlights the most related lines of research.

II. Related Work

This section explores initiatives related for IBIDAE from a triple perspective: i) an architectural perspective, ii) efficient map-reduce models, and iii) different industrial benchmarks for industrial machines. In all possible cases, the architecture is compared against other similar industrial approaches to establish a nexus between current industrial initiatives and IBIDAE.

A. Big-Data Architectures

HDP (HortonWorks Data Platform) [30] is a general purpose platform for data processing. Its goal is to integrate tools from different vendors. It integrates all different big-data technologies (HDFS, Spark, Storm, or other Apache projects). These technologies are available for developing different projects, which are integrated by vendors. The hierarchical approach of IBIDAE may be integrated as another technology with enhanced primitives. Also, the idea of splitting an application into different parallel segments, proposed by IBIDAE, running on partially isolated clusters, is valuable to extend the engine of HDP.

Oracle has its own architecture for big-data and incorporates open-source technology [31]. In essence it includes the same type of tools described in the HDP platform (i.e. a set of open-source Apache projects). From the perspective of the Oracle architecture, the hierarchical model proposed in IBIDAE is a mechanism to improve performance, which may be included as an enhanced architecture.

The lambda architecture derivates all data to a batch layer that stores information and also to a speed layer that prompts data that are accessed via a service layer (see [14]). The idea in the lambda architecture is to be able to use batch and online processing at the same time. Basically, batch is supported by map-reduce engines for offline batch processing and distributed stream processors for online data processing. One technology used to implement map-reduce is Spark and/or Hadoop. Therefore, the hierarchical techniques proposed for IBIDAE, which runs on Spark, are directly useful to speed up the map-reduce nodes described in the lambda architecture.

An industrial big-data ingestion and analysis platform for industrial systems (IBDP) integrates a stack for industrial analytics with cloud computations [25][29]. To offer this type of facility, the architecture integrates a classical stack for big-data processing: HDFS, Spark, and open-stack for cloud computation. IBDP copes with three different types of industrial big-data: thick data, which refer to business relevance; fast data which need a prompt response; and slow data which need response times in the range of hours. The architecture proposed for IBIDAE, running on Spark and Hadoop, may be integrated with the IBDP engine to increase performance or reduce the number of required resources for a given analytic.

Decathlon [24] is an industrial platform that offers modern software that helps automation industry to achieve their business objectives. It offers a set of modules that includes a big-data engine which is useful to take business decisions based on for example condition monitoring or predictive maintenance results. From an industrial perspective, IBIDAE may be also beneficial to Decathlon to speed up performance as the data to be analyzed increases.

None of the previous products or architectures (HDP, Oracle, IBDP, lambda architecture and the Decathlon industrial) has integrated the hierarchical model within their cores. Current engines are based on plain map-reduce models that are composed of a single cluster of machines that share resources. In this context IBIDAE contributes a hierarchical map-reduce computational model with multiple clusters and operations running in parallel.

B. Map-Reduce Optimizations

Several researchers analyzed the performance of map-reduce. For instance [19] analyzes the set of configuration parameters that may have an impact on the performance, developing a benchmarking configuration method to determine the set of configuration parameters that minimize the execution time of the map-reduce interaction. [19] offers a subset of them that has impact on performance, being able to reduce execution times in 32%. The proposed hierarchical model also adds an alternative way to reduce execution time in applications, by parallelizing their execution in multiple engines that share data. In the evaluation, test-bed reductions may reach 60% of the total time for some configurations with intensive I/O operations. The optimization algorithms described in [19] may be extended with the proposed hierarchical model in IBIDAE. In this case IBIDAE requires to be optimized to meet application needs.

Researchers have proposed a concurrent map-reduce [20] that deals with applications that have to share large amounts of data, improving performance by a factor of 4. The proposed hierarchical approach generates higher speedup factors (130x) using a hierarchical approach that parallelizes the execution of an analytic. However, both techniques differ in the way they accomplish their goals. IBIDAE includes a concurrent model to split the execution in a hierarchical model, as it is done in a operating systems with processes and tasks, having different map-reduce activities (of the same application) running in

parallel. On the other hand [20] is focused on the optimization of reduce phases to increase performance.

Lastly, there is a last relevant optimization for skewed map-reduce applications, named FP-Hadoop [32], which incorporates intermediate reducers to offer improved performance. This optimization is able to offer an improvement of a factor of 5 in total execution time. IBIDAE is able to offer higher speedup factors.

C. Big-Data Benchmarks

Currently, there are a number of big-data benchmarks that may be used to compare equivalent technologies [34]. Some of them also may generate synthetic benchmarks. One of them is the BigdataBench [34], which is based on a suite for Internet services. Among the different workloads available, the proposed industrial service belongs to the sort and word-count micro-benchmarks.

The benchmark used in IBIDAE comes from an industrial surveillance domain. It is similar in size to other similar approaches from the industrial scenarios (see [3], [26], and [27]). The benchmark processes surveillance logs stored by a set of 1600 sensors spread out in factory subsystems, generating alarms which are processed later. The benchmark processes logs with 0.1 Gbytes, 1 Gbytes, and 10 Gbytes of data. In the logs each line of the file corresponds to the description of an event with the following relevant information: 1) Place where the sensor is located: text with a description; 2) Date: text representing a day; 3) Event generated: textual description of the event; 4) Relevance of the event (medium, high, low); 5) Monetary costs; and 6) Speed of the detected event. In comparison, BigDataBench datasets are larger than in the industrial surveillance application defined for IBIDAE, which comes from a real surveillance trace.

Another benchmark is the TPCx-HS that adds formal specification and enforcement rules that enable comparison of results among systems [35]. One characteristic of TPCx-HS is that it follows a stepped scaled sizing model on BigdataBench. From the perspective of the benchmark used in IBIDAE the type of surveillance application used corresponds to a new application in BigdataBench.

Finally BigBench [36] offers an end-to-end benchmark approach that addresses variety, velocity, and volume aspects of many big-data systems in the analytic domain. It also reports the generation and exploration of 200 Gbytes of data. Analytics specified in BigBench offer response times in few seconds to less than one hour. Comparatively, our benchmark is smaller in size (0.1 Gbytes, 1 Gbytes, and 10 Gbytes) than BigBench.

NASA's repository [36] also offers an access to eleven data-sets ranging from few hundreds of bytes to 3 Gbytes of data. IBIDAE offers a larger amount of data: up to 10 Gbytes from a simple a surveillance application.

III. IBIDAE

IBIDAE defines an architecture for industrial applications that supports hierarchical efficient map-reduce processing. The architecture is partially based on general purpose and other big-data architectures such as the lambda architecture [14], a cyber-architecture for real-time industrial applications described in [3], and another architecture described for cloud computing in [22]. From [3] IBIDAE derived an industrial information subsystem in charge of connecting factory floor elements to populate a big-data repository. From [22] it takes the idea of processing offline and online data, stored in a large storage space. Finally, from [14] it derived the idea of being compatible with the lambda computational model.

A. Architecture

The basic map-reduce industrial architecture of IBIDAE (see Fig. 1) consists of three layers in a top-down approach:

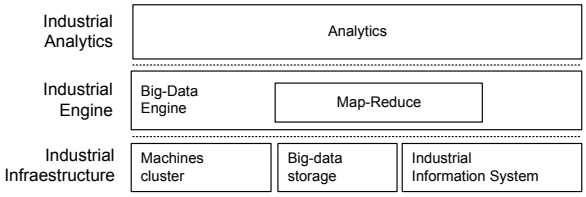

Fig. 1. Industrial Big-Data Architecture

- *Industrial analytics layer*: This layer is in charge of performing industrial analytics. Analytics are processed by the engine that uses the resources of a map-reduce infrastructure layer to speed up computations and access data.
- *Industrial engine*: Basically, it is in charge of coordinating different aspects of the big-data ecosystem. It is in charge of providing proper support to provide distribution and parallelism via a map-reduce engine, offering support to different analytics. Typical stacks include Hadoop, Storm, and Spark technologies. The main difference in an industrial system is probably the type of application they have to take into account.
- *An industrial infrastructure layer:* It provides low-level resources stored in a cluster of machines. Typical clusters [17][28] include MESOS, standalone clusters, and YARN support. To store data, it includes a HDFS (Hadoop Distributed File System) filesystem that offers safe access to large datasets stored in it. It may also refer to any other distributed filesystems (e.g. NFS or Lustre) able to offer large and efficient storage space.

In addition to the basic infrastructure, there is a hierarchical approach used to increase scalability in different models (Fig. 2). Basically, the hierarchical infrastructure splits resources into different units to improve performance. The approach assumes that in some cases having a single space view is much more inefficient than splitting this model to offer improved performance.

This hierarchical approach impacts on all elements of the industrial architecture ranging from the clusters used for storage to execution engines that support analytics. In all cases, the application splits data into a set of *m*-tasks that can be executed in *n*-different elements of the architecture. This type of approach is able to increase parallelism as resources may be used more efficiently in parallel to give improved performance, as well as to reduce the number of messages

required to perform analytics.

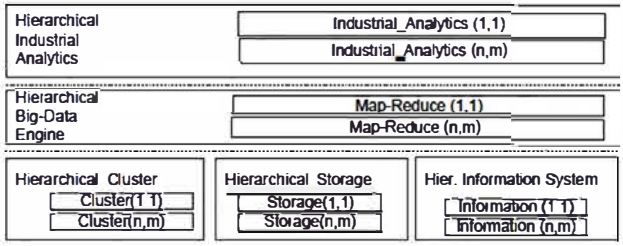

Fig. 2. Hierarchical Industrial Big-Data Architecture

Potentially, the idea of splitting or partitioning resources to increase performance is applied to all elements of the architecture:

- On the analytic layer: Many analytics may be arranged as a set of hierarchical subtasks where the results of a subtask are used to develop different types of analytics. The application should be able to provide a hierarchical organization in accessing resources. Therefore, the analytic should be able to be split in different subtasks easily, requiring in some cases help from the developer to provide this hierarchical aspect.
- On the big-data layer: Although many of them provide plain map-reduce models (based on traditional map-reduce primitives), they lack direct support for implementing the hierarchical model as a part of their logic. The map-reduce already includes the possibility of running different subtasks in an analytic. This support should be extended with the possibility of using multiple clusters.
- On resources layer: Clusters may offer a hierarchical organization as they are properly separated into different computational units. Many storage spaces provide hierarchical access by means of partitioned resources. Also, in many cases, the information available from the industrial information system may be accessed with different levels of quality, referring information from a specific machine of a certain floor or global information about business. Therefore, the idea is to identify the mechanisms that enable sharing data among different nodes to offer an efficient configuration.

The main advantage of the hierarchical approach against the plain map-reduce is that it may split work into smaller pieces that may be more efficiently processed in parallel. This is of special interest in applications that have high processing costs (e.g. an expensive n^2 sort process) that can see how performance is sped up with a hierarchical approach. For instance, the use of a simple hierarchy of ten elements may reduce computational costs in 10 times splitting data. Another advantage is that peaks in computations get reduced with a hierarchical approach that works with a reduced amount of data.

B. Computational Model

The basic computational model of a map-reduce interaction consists of a map functions that takes an input key and a value and outputs a list of elements, each of them with a key and a value:

```
map(Key1, Value1)-> List(Key2, Value2)
```
(eq. 1)

After that reduce functions receive this list of elements for each key and produce an output:

```
reduce(Key2, List (Value2))-> List(Value3)
```
(eq. 2)

The hierarchical map-reduce (`h_map`) model adds parallel execution for a set of resources that are reduced to a list of values. Internally, the hierarchical map is composed of a set of map-reduce that run in parallel:

```
h_map(Key1, Value2,m)-> List(Key2, Value2)
  (1) || map(key, value))-> List(Key_1,Value_1)
  (.) || …
  (m) || map(key, value)->List(Key_m,Value_m)
      reduce_h(key,List(Value))-> List(Key2, Value2)
```
(eq. 3)

This new type of primitive may be implemented as a set of map-reduce operations that run in parallel (`||`) in order to speed up execution and perform better resource utilization. They are followed by an specific reduce phase (`reduce_h`) that converts this m-lists into a list of values each one of them generated as a result of a map-reduce interaction into a list of values. In addition, it allows defining a number of n-clusters that receive and process in m-parallel data units. Each of these clusters process part of the m-tasks in parallel using an `h_map` function of `m/n` elements. The resulting hierarchical equation that includes clustering is:

```
h_map(Key1, Value2,n,m)-> List(Key2, Value2)
  (1) || h_map(key, value, m/n))-> List(Key_1,Value_1)
  (.) || …
  (n) || h_map(key, value, m/n)->List(Key_m, Value_m)
      reduce_h(key, List(Value))-> List(Key2, Value2)
```
(eq. 4)

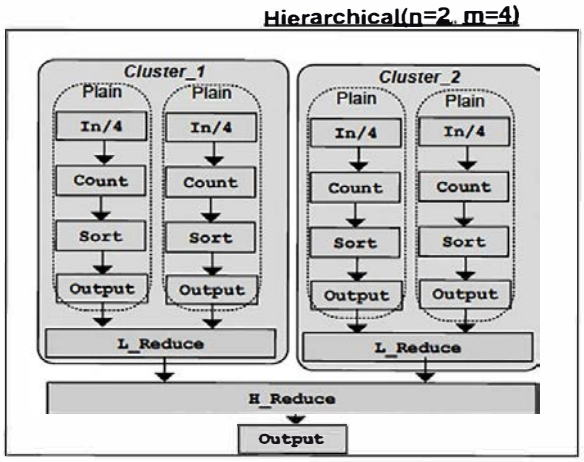

Fig. 3. Hierarchical map-reduce in a simple `count_and_sort` application. The hierarchical approach creates new tasks that execute in parallel and are able to process data in a hierarchical fashion. The model requires two additional `reduce` functions to mix the outputs of each computation, in an `L_Reduce` phase and globally with an `H_reduce` phase. In the case of an `L_Reduce` stage the idea is to combine ordered lists. The same idea persists in the case of an `H_Reduce` stage.

As in the previous case (see eq. 3), there is a reduce phase in charge of reducing data after performing n-hierarchical map functions. To illustrate an application that benefits this computational model, it was extended the `wordcount` analytic, which has been modified to include a sort phase (see

Figure 3).

C. Impact on Current Tools

For the integration of the techniques in existing technology, we observe two different ways: a non-invasive one and an invasive one that requires modification inside current big-data tools.

The non-invasive approach replicates elements of the architecture in different places to be compliant with the hierarchical model. The advantage of the non-invasive approach is that current technology may be easily used without any modifications in their cores.

The invasive approach consists in extending current tools to take into account the hierarchical model while keeping a unified system view of the programming model. For instance, in the case of the RDD model of Spark [23], a solution to integrate the hierarchical map-reduce is to extend the programming model with `h_map` functions in the API with these functions:

```
h_map(map_fun, reduc_fun, par_units)                (fun. 1)
h_map(map_fun, reduc_fun, cluster, par_units)       (fun. 2)
```

The first function takes as an input a map function (`map_fun`), its internal reduction function (`reduc_fun`), and the number of parallel units (`par_units`) to be executed before calling the reduce function. Internally, it performs the computation described in Equation 3. The second function adds information about the number of clusters (`cluster`) required to process the analytic and refers to Equation 4. Our current implementation uses the non-invasive approach as it is simpler from the perspective of an initial implementation.

IV. Empirical Evaluation

A. Prototype Description

A prototype of IBIDAE has been developed to evaluate the performance and the benefits stemmed from the use of a hierarchical approach against a traditional stack (see Table I). For the evaluation we built a system with a maximum of 64 low-cost machines interconnected with 2 Gbytes network. Extending Hadoop 2.6.1 and Spark 1.6.1, we built the engine of IBIDAE (see Figure 4) named H-Spark_1.6.1. The deployment engine instantiates one H-Spark engine per machine. To check the performance, we derived a benchmark from a surveillance application described in the related work section, which has been used to evaluate the performance offered by IBIDAE. The goal of the evaluation is to establish empirical evidence on the speedups one may expect from the combination of these hierarchical and parallel models as a trace is processed.

Our empirical results (see Table I) show an important speedup experienced in the application with the use of a hierarchical approach; using the same number of machines hierarchical map-reduce outperforms a plain map-reduce solution. The maximum performance of the proposed setup refers to a speed-up factor of 130x for the hierarchical map-reduce, and a minimum of 8x for plain map-reduce models.

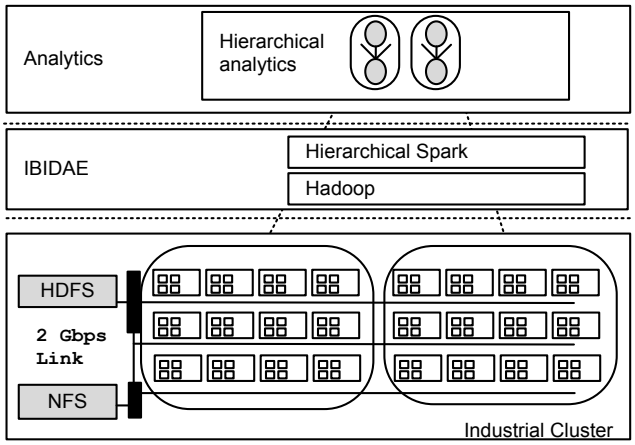

Fig. 4. Stack used in the evaluation

Table I: Cluster, software stack, and benchmark application characterization

Cluster	
Resources Available	64 machines with 256 cores (1,1 GHz) 512 GB of RAM- 5120 GB HD
Machine features	4 cores, 8 GB of memory and 80 GB local disk space (per machine)
Network	2 GBPS optical fiber accessed via LANs
Software Stack	
Hadoop	2 6 1
Spark	1 6 1
H- Spark	1 6 1-hierarchical
Surveillance Benchmark	
Industrial application	Process data and generated a list with: IA1: Most popular sensors; IA2: Less active sensors; IA3: Most popular events; IA4: Less popular events; IA5: Highest speed detectors; IA6: Lowest speed detectors; IA7: Most relevant alarms; and IA8: Less relevant alarms
Analyzed Data Log	Small: 0,1 Gbytes: 10 partitions (blocks) Normal: 1 Gbytes: 100 partitions Large: 10 Gbytes: 1000 partitions
Outcome for (small, normal, and large) data	
Total time:	[70 sec; 10 min; 1.8h]
Plain M-R speedup:	[8x; 9x; 10x]
Plain H-M-R speedup:	[110x; 117x; 130x]

B. Performance Patterns

1) Plain Map-Reduce Approach

To evaluate the performance of the infrastructure, first we evaluated the influence of adding more machines to reduce response time in all analytics. For this purpose we measure the different speedups reached when adding additional cores to the system. In the proposed ecosystem, each time we add new machines, the total time required to execute the application reduces, and the speed (defined as $speed = \frac{}{}$) increases. Ideally, this factor should grow linearly with the number of cores available in the system. However, due to different characteristics of the infrastructure, its growing tends to be as follows: there is a i) first phase where infrastructure overhead rules, and adding cores increases more than linearly in performance because this time is masked by the overhead of the infrastructure; ii) a second one, where adding cores reduces proportionally computation times, and where the cluster work is dominant; iii) a third stage where adding more cores does not linearly increase performance anymore due to the overhead caused by communications among the different

elements of the cluster. Fig. 5 shows this evolution for the set of proposed analytics. Results show the maximum speedup factor is 6.7 for a system with 64 cores compared to a single core setup. The example also shows that phase i) is from 1 to 4 cores, phase ii) is from 4 to 32 cores, and phase iii) is with setups that have more than 32 cores.

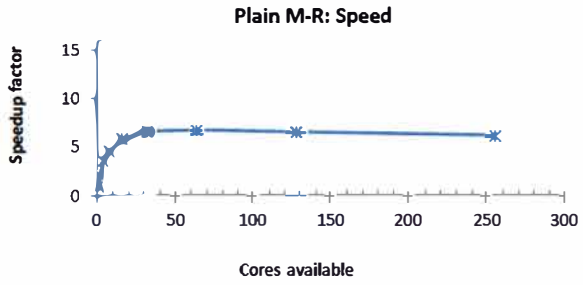

Fig. 5. Speedup factor offered adding cores. The figure refers to the cost of running all eight analytics with 1 Gbytes of data.

Also of great importance is the efficiency associated to adding new cores to the system (Fig. 6). The efficiency is defined as the speed divided by number of cores used to implement the system, i.e.: $efficiency = \frac{speed}{cores}$. Ideally, i.e. assuming that all cores are able to reduce the response time proportionally, efficiency should be a constant function. However, the empirical pattern obtained in our experiments show how the normalized efficiency decreases with number of cores available from a normalized 1, which corresponds to a single core, to a minimum of 4% with an industrial big-data engine that runs 256 cores (i.e. 64 machines). The main cause for the inefficiency is that in some stages of the analytics, the number of blocks (partitions in the Spark jargon) is less than one hundred and at least it should be equal to the number of available cores.

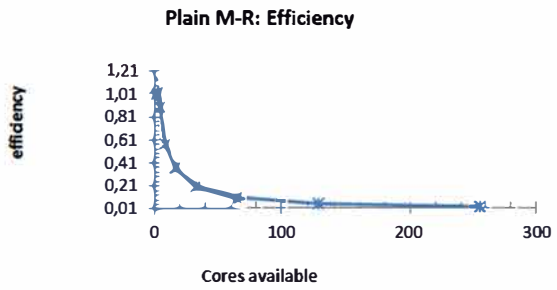

Fig. 6. Efficiency related to the scenario described in Fig. 5.

The second source of inefficiency is that all cores are blocked for the analytics. This type of effect has a negative impact in performance as the code performs many I/O operations that do not use all available resources for running the analytic. From the point of view of performance, an hierarchical approach may deliver higher performance numbers, as analytics may run different parts of the application more efficient in smaller clusters, which are a priori much more efficient than larger ones. For the analyzed setup, the advantages offered by systems with more than 64 cores are marginal and it is preferable, from the perspective of performance, to run analytics concurrently (as shown in Figure 5 and Figure 6).

2) Hierarchical Map-Reduce Patterns

For the hierarchical approach, the cluster is split in different parallel map-reduce applications which are combined later, using the hierarchical map-reduce model proposed in IBIDAE. This idea, similar to the idea of multiprocessing, is to be able to improve performance in inefficient scenarios, creating smaller clusters that increase performance.

The experiment introduces results for an application which is split into two, four, and eight different analytics, whose results have to be combined as they finish their executions. The example also adds the possibility of using only one of both clusters. Results (see Fig 7) for the absolute speedup due to the use of hierarchical approach with clusters of 1-32 cores per cluster show that the technique is able to offer acceptable performance for the analytics carried out (8 stages with 1 Gbytes of industrial data logs). As in the previous experiment, the speedup factor degrades as the number of cores in the cluster increases. For the given configuration, with eight different applications, in the hierarchical model the maximum speedup is higher than the speedup offered by the plain map-reduce model due to the blockings suffered in the plain map-reduce model. Results show that the speedup factor increases with number of analytics in the core, and decreases with number of nodes in cluster, because there is not enough partitions to be processed. This behavior can be seen along the vertical axis of Fig 8.

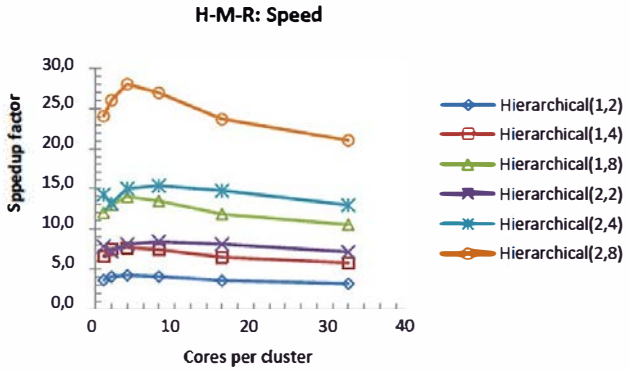

Fig. 7. Speedup factor offered by the hierarchical approach

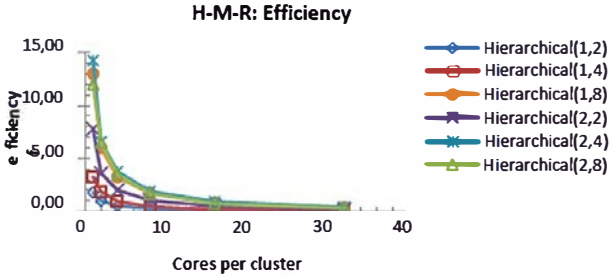

Fig. 8. Efficiency offered by the hierarchical approach

Efficiency results (see Fig. 8) show a decreasing efficiency with an increasing number of available cores per cluster as in

the previous experiment. But results show that in efficiency terms the use of a hierarchical approach always increases performance because it reduces the number of partitions to be processed in each node. Also, the use of two clusters instead of one is also beneficial in terms of performance because it parallelizes execution.

3) Plain M-R vs. H-M-R Patterns

This experiment compares plain map-reduce against its hierarchical approach. Results in terms of speedup are shown in Fig. 9 and Fig. 10 refers to efficiency results. Results show that the use of a hierarchical approach is beneficial for the evaluated scenarios outperforming plain map-reduce models. In all scenarios, the hierarchical approach is able to increase the speed of the system.

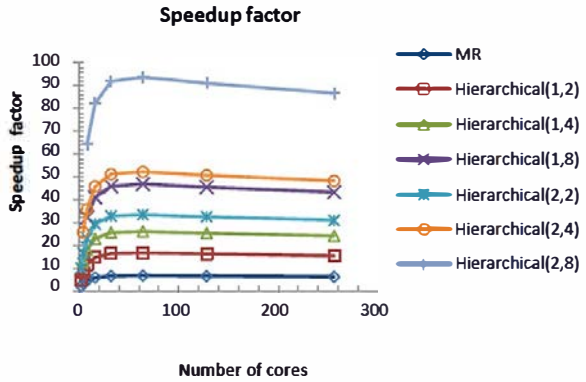

Fig. 9. Comparing speedup factors

Notice as the speedup for the plain map-reduce model is below all speed curves for the same amount of work (see Fig. 9). Speed curves for two, four and eight split applications show remarkable increase in speed processing, which probably is the main benefit stemmed for the use of a hierarchical approach. The gap among the plain M-R and the H-M-R is 14 times as 2 clusters and 8 segments are used to produce the hierarchical application (see Figure 9).

In terms of efficiency (see Fig. 10) and comparing plain map-reduce (MR) with an application composed of two (2), four (4), and eight (8) parallel analytics, the empirical suggests that the use of a hierarchical approach increases efficiency because it produces smaller clusters. The use of two clusters also improves efficiency in comparison with a single cluster machine.

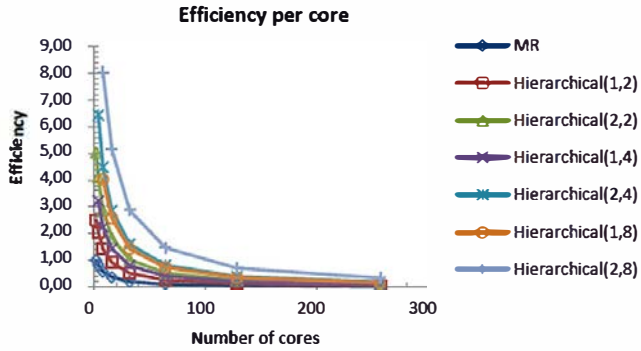

Fig. 10. Combined efficiency

C. Benchmark Performance

This section deals with the evaluation of the full benchmark, which includes data sets ranging from small 0.1 Gbytes to large amounts of data: 10 GBytes. This experiment extends previous micro-benchmarks to the surveillance benchmark for the speedup and efficiency of the plain map-reduce and hierarchical map-reduce models.

Regarding speedup factors (see Fig. 11), results show that an increase in the amount of data has a positive effect in the maximum achievable speed (130x). This increase in speed is due to the technological overhead of the approach which surpasses communication costs. Results also show how a huge amount of data reduces maximum speed. This reduction is mainly due to an increase in the number of partitions, which potentially enable a higher number of cores to be used.

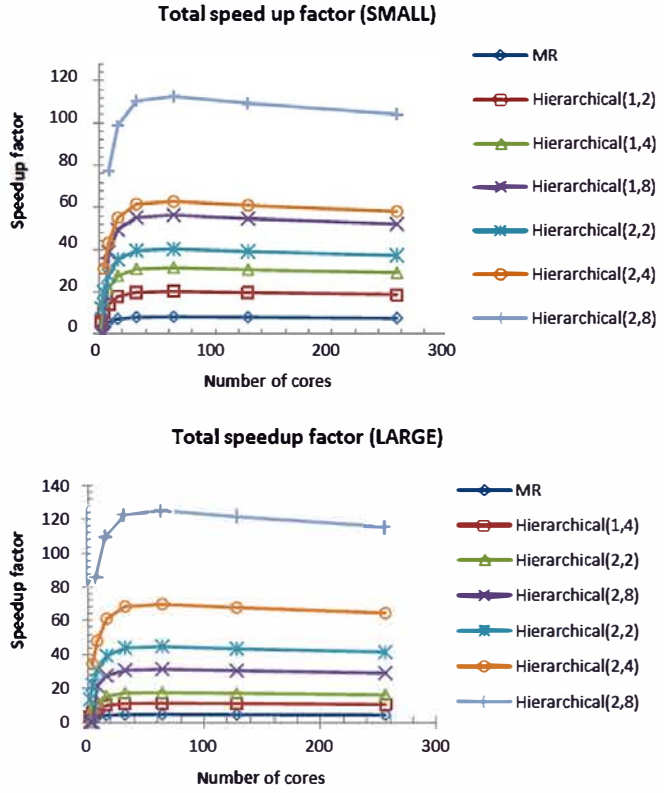

Fig. 11. Compared speedup with SMALL (0.1 Gbytes) and LARGE (10 Gbytes) amounts of data

Likewise, Fig. 12 describes efficiency one may achieve with the proposed models for 0.1 Gbytes of data, and 10 Gbytes. Results corroborate previous performance terms and show how the use of a hierarchical approach may improve efficiency. As in the previous cases, performance gets reduced as the number of cores increases because the application is more inefficient.

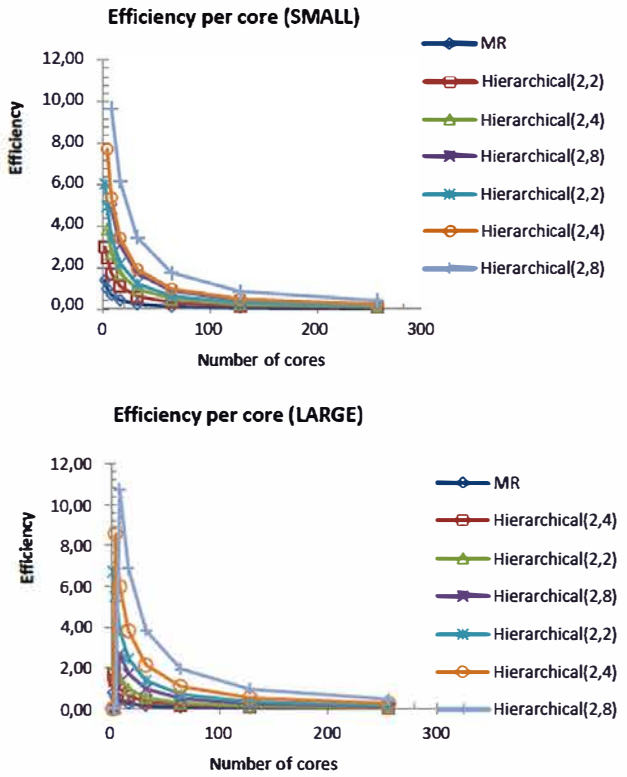

Fig. 12. Compared speedup with SMALL (0.1 Gbytes) and LARGE (10 Gbytes) amounts of data

The main conclusion drawn from the evaluation is that the hierarchical approach is able to increase performance remarkably (up to a maximum of 130x). Results are in line with what happens in parallel systems where the use of clustering (hierarchical approaches) versus flattering computation (plain models) is beneficial from the perspective of performance. The obtained results suggest that future industrial architectures should consider the use of hierarchical approaches to offer scalable performance. In comparison to a single M-R entity, H-M-R increases performance due to two complementary effects. The network bottleneck and serialization/deserialization issues disappear because computation may run in isolated clusters. The second is that the cost of data representation is small as the data is split in Spark.

V. Conclusions And Future Work

Next generation of industrial system will be part of a global big-data infrastructure able to provide self-configuration to take intelligent decisions to operate efficient businesses. In this context, the proposed architecture increases scalability by means of a hierarchical approach that improves current map-reduce performance, increasing performance of current map-reduce models. As a result of a proper hierarchical organization, the system may see how performance increases, with a large speedup factor. The empirical evidence showed how the performance of the engine may be improved by splitting resources among different analytics to speedup performance.

Our current research efforts are extending current machine learning engines based on Spark with the hierarchical approach to increase operational performance of general purpose applications. Also, we are extending the proposed techniques to larger cluster infrastructures equipped with enterprise machines. Finally we explore the integration extending results given in [26] and [44-47] for distributed stream processing as a building block for IBIDAE.

Acknowledgements

Work partially supported by "Distributed Java Infrastructure for Real-Time Big-data" (CAS14/00118), eMadrid (S2013/ICE-2715), HERMES-SMARTDRIVER (TIN2013-46801-C4-2-R), and AUDACity (TIN2016-77158-C4-1-R). We thank our anonymous reviewers their efforts in improving the quality of the article providing suggesting over 300 changes in the original article.

References

[1] J. Lee, H. Kao, S. Yang "Service Innovation and Smart Analytics for Industry 4.0 and Big Data Environment". Procedia CIRP, Volume 16, 2014, Pages 3-8, ISSN 2212-8271.

[2] L. Jay, B. Bagheri, H. Kao "Recent advances and trends of cyber-physical systems and big-data analytics in industrial informatics". International Conference on Industrial Informatics (INDIN). July, 2014 pp.1-2.

[3] E. P. Xing et al. "A New Platform for Distributed Machine Learning on Big Data". IEEE Transactions on Big Data 1(2) pp. 49-67 (2015)

[4] P. Basanta-Val, M. García-Valls "A Distributed Real-Time Java-Centric Architecture for Industrial Systems". IEEE Trans. Industrial Informatics 10(1) pp. 27-34 (2014)

[5] Y. Shen, O. Kaynak "Big-data for Modern Industry: Challenges and Trends [Point of View]". Proceedings of the IEEE 103.2 (2015) pp. 143-146

[6] O. Marek, V. Jirkovský, J. Bezdíček. "Big-data challenges in industrial automation". Industrial Applications of Holonic and Multi-Agent Systems. Springer Berlin Heidelberg, (2013) pp. 305-316

[7] J. Williams, K. S. Aggour, J. Interrante, J. McHugh, E. Pool "Bridging high velocity and high volume industrial big-data through distributed in-memory storage & analytics". 2014 IEEE International Conference on Big Data (Big Data), Washington, DC, 2014, pp. 932-941

[8] X. Wu, X. Zhu, G. Q. Wu, W. Ding "Data mining with big-data". IEEE Transactions on Knowledge and Data Engineering, 26.1 (2014) pp. 97-107.

[9] C. Hsinchun, R. H. Chiang, V. C. Storey "Business Intelligence and Analytics: From Big-data to Big Impact". Journal MIS quarterly 36.4 (2012) pp. 1165-1188.

[10] J. M. Tien "Big-data: Unleashing information." Journal of Systems Science and Systems Engineering 22.2 (2013) pp. 127-151.

[11] A. J. Guzzo et al. "Big-data recommendations for industrial–organizational psychology". Industrial and Organizational Psychology 8.04 (2015) pp. 491-508.

[12] M. Minelli, M. Chambers A. Dhiraj "Big-data, big analytics: emerging business intelligence and analytic trends for today's businesses". John Wiley & Sons, 2012. ISBN: 978-1-118-14760-3

[13] J. Rifkin "How the Third Industrial Revolution Will Create a Green Economy". New Perspectives Quarterly 33.1 (2016) pp. 6-10.

[14] N. Marz and J. Warren "Big-data: Principles and Best Practices of Scalable Realtime Data Systems" (1st ed.). Manning Publications Co., Greenwich, CT, USA. 2015. ISBN: 9781617290343
[15] K. Shvachko, H. Kuang, S. Radia, R. Chansler "The hadoop distributed filesystem". In IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST) pp. 1-10.
[16] Storm "Distributed and fault-tolerant real-time computation". Available (2016) on https://storm.incubator.apache.org/
[17] Spark "Lightning-fast cluster computing". Available (2016) on https://spark.apache.org
[18] S. Pandey, V. Tokekar "Prominence of MapReduce in big-data processing". 2014 Fourth IEEE CSNT pp: 555-560
[19] J. Kim, T. K. A. Kumar, K. M. George, N. Park "Performance evaluation and tuning for MapReduce computing in Hadoop distributed file system". IEEE 13th INDIN, Cambridge, 2015, pp. 62-68.
[20] F. Zhang, M. Q. Malluhi, T. M. Elsyed. "ConMR: Concurrent MapReduce Programming Model for Large Scale Shared-Data Applications". In Proceedings Conference on Parallel Processing (ICPP '13) pp. 671-679.
[21] S. Pandey, Vrinda Tokekar. "Prominence of MapReduce in big-data processing". Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference on. IEEE, 2014 pp.33-38
[22] C. Wang, B. Zhuming, L. D. Xu "IoT and cloud computing in automation of assembly modeling systems." IEEE Transactions on Industrial Informatics, 10.2 (2014) pp. 1426-1434.
[23] M. Zaharia et al. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing". Proceedings of the 9th USENIX, (2012) pp: 2-2
[24] ABB, Decathlon. Available online (2016) at http://new.abb.com/docs/librariesprovider9/about-us-fr---data-centers-files/abb-decathlon.pdf?sfvrsn=4
[25] C. Ji, S. Liu, C. Yang, L. Wu, L. Pan "IBDP: An Industrial Big-data Ingestion and Analysis Platform and Case Studies" 2015 International Conference on (IIKI), Beijing, 2015, pp. 223-228.
[26] P. Basanta-Val, N. Fernández-García, A. J. Wellings, N. C. Audsley "Improving the predictability of distributed stream processors". Future Generation Comp. Syst. 52 pp. 22-36 (2015)
[27] P. Basanta-Val, M. García-Valls "Resource management policies for real-time Java remote invocations". J. Parallel Distrib. Comput. 74(1) pp. 1930-1944 (2014)
[28] K. Kambatla et al. "Trends in big-data analytics." Journal of Parallel and Distributed Computing 74.7 (2014) pp. 2561-2573.
[29] Ji, Cun, et al. "Device Data Ingestion for Industrial Big-data Platforms with a Case Study." Sensors 16.3 (2016): 279.
[30] Hortworkworks, "HortonWorks Data Platform". Available online (2017) at http://hortonworks.com/products/hdp/
[31] Oracle, "Oracle Big-data reference architecture" Available online (2017) at http://www.oracle.com/technetwork/topics/entarch/oracle-wp-big-data-refarch-2019930.pdf
[32] M. Liroz-Gistau et al. "FP-Hadoop: Efficient processing of skewed MapReduce jobs". Information Systems 60 (2016) pp. 69-84.
[33] M. Paiva Ramos et al."Distributed systems performance for big-data". Information Technology: New Generations. Volume 448 of the series Advances in Intelligent Systems and Computing pp. 733-744
[34] L. Wang et al. "Bigdatabench: A big-data benchmark suite from internet services". IEEE 20th International Symposium on High Performance Computer Architecture (HPCA) (2014) pp. 80-92.
[35] R. Nambiar et al. "Introducing TPCx-HS: The First Industry Standard for Benchmarking Big-data Systems". TPCTC 2014, LNCS 8904, pp. 1–12, 2015. DOI: 10.1007/978-3-319-15350-6_1
[36] Ghazal et al. "BigBench: towards an industry standard benchmark for big-data analytics". In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data pp. 1197-1208.
[37] NASA Prognostics Center of Excellence (PCoE). "PCoE Datasets". National Aeronautics and Space Administration. Available online (2017) at http://ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository/
[38] P. Basanta-Val, et al. "Architecting Time-Critical Big-Data Systems," in *IEEE Transactions on Big Data*, vol. 2, no. 4, pp. 310-324, Dec. 1 2016.
[39] J. Lee (2015). Industrial Big Data. China: Mechanical Industry Press. ISBN 978-7-111-50624-9.
[40] P. Basanta-Val, N. Fernández-García, L. Sánchez-Fernández, "Predictable remote invocations for distributed stream processing", In Future Generation Computer Systems, 2017, ISSN 0167-739X, https://doi.org/10.1016/j.future.2017.08.023.
[41] P. Basanta-Val, N. Fernandez-Garcia, L. Sanchez-Fernandez and J. Arias Fisteus, "Patterns for Distributed Real-Time Stream Processing," in *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, no. 99, pp. 1-1.doi: 10.1109/TPDS.2017.2716929
[42] Zhihan Lv et al.: "Next-Generation Big Data Analytics: State of the Art, Challenges, and Future Research Topics". IEEE Trans. Industrial Informatics 13(4): 1891-1899 (2017)
[43] Mariluz Congosto, Pablo Basanta-Val, Luis Sanchez-Fernandez, "T-Hoarder: A framework to process Twitter data streams, In Journal of Network and Computer Applications", Volume 83, 2017, Pages 28-39, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2017.01.029
[44] P. Basanta-Val, M. García-Valls and I. Estévez-Ayres, "Towards a Cyber-Physical Architecture for Industrial Systems via Real-Time Java Technology," 2010 10th IEEE International Conference on Computer and Information Technology, Bradford, 2010, pp. 2341-2346.
[45] Pablo Basanta-Val, Marisol García-Valls: "A library for developing real-time and embedded applications in C". Journal of Systems Architecture - Embedded Systems Design 61(5-6): 239-255 (2015)
[46] M. Teresa Higuera-Toledano: "Java Technologies for Cyber-Physical Systems". IEEE Trans. Industrial Informatics 13(2): 680-687 (2017)
[47] M. V. Moreno *et al*., "Applicability of Big Data Techniques to Smart Cities Deployments," in *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 800-809, April 2017. doi: 10.1109/TII.2016.2605581