Industrial Analytics Pipelines

K. Eric Harper, Jiang Zheng, Sam Ade Jacobs, Aldo Dagnino, Anton Jansen
Thomas Goldschmidt, Adamantios Marinakis
ABB Corporate Research
Raleigh, NC USA; Västerås, Sweden; Ladenburg, Germany; Baden-Dättwil, Switzerland
{eric.e.harper, jiang.zheng, sam.jacobs, aldo.dagnino}@us.abb.com, anton.jansen@se.abb.com,
{thomas.goldschmidt@de.abb.com, adamantios.marinakis@ch.abb.com

Abstract— Decreasing cost and increasing capabilities of instrumentation, networks and data repositories have pervaded the industrial automation and power markets and opened the door for large scale collection and analysis of data. There are a variety of technology stacks that can be applied to these types of activities. However, no single infrastructure or architecture fits all the scenarios. With limited data science training and experience, it is difficult and time consuming for highly specialized domain experts to choose the optimal approach. In this paper, we introduce an architectural pattern for the design of a flexible core analytics platform which is extensible using different pipelines. The pipeline pattern provides an accelerated start to implementing industrial analytics applications. The platform enables domain experts to compose pipelines in series and in parallel at scale with the right quality attribute trade-offs to deliver significant business value. Our use of the proposed platform is illustrated with real-world industrial applications, which necessitate various data handling and processing capabilities. These examples show the importance of the platform to non-data experts: reducing the learning curve for applying data science, providing a systematic rating process for choosing the pipeline types, and lowering the barriers for industrial businesses to leverage analytics.

Keywords— industrial analytics; data science; architecture patterns; product line architectures

I. INTRODUCTION

Industrial companies take pride in advanced engineering: no one else knows the detailed design and operation of their products, or is better suited to analyzing measured data collected under laboratory and field conditions. Insights come from system models and the physics, mechanics, and dynamics of the component interactions. Better yet, the engineers have extensive experience solving customer issues which lead to product refinements and enhancements.

Many engineering models are approximations, for example linear simplifications of non-linear behavior. Approximations can be validated using statistical models. This is especially true in reliability engineering [1] and process control [2]. Modern statistics depends on IT resources: computer databases and processors that can store and evaluate large amounts of data. Statistics in non-engineering domains has evolved into the fourth paradigm [3]: data-intensive scientific discovery. Known as data science, this discipline incorporates exploration of data and construction of technology stacks for delivering business value.

Application of data science in industry brings together operational (OT) and information (IT) technology platform stakeholders, all anticipating the potential for generating significant business value. Each domain is invested in its own architectures with accepted standards and common practices. Choosing the best software architecture depends on identifying and addressing the non-functional requirements, and this creates common ground for stakeholders to design industrial analytics applications.

Industrial analytics is the intersection of data science and software architecture. Industry has reported success with targeted applications [4] where data science has been employed for specific analytics tasks. An implementation provides business value in one scenario, but the solution in another application domain might require a completely different approach to be realized optimally.

A variety of analytics approaches have emerged for defined types of tasks commonly referred to as “big data”. Different initiatives in the open source community have emerged in recent years. Distributions of open source software [5] combine these implementations as comprehensive technology stacks.

Market signals regarding industrial analytics and strong encouragement from our senior management put our team on a multi-year journey to investigate and review analytics alternatives, and then apply our knowledge to application domains benefiting our businesses and customers. The main contributions of this paper are an architectural pattern allowing diverse technology stacks to be composed together in a reusable way and an approach for choosing the right technology stack for industrial analytics.

The rest of the paper is structured as follows: Section II presents related work, Section III introduces the concept of analytics pipelines. Section IV describes our approach for choosing the optimal analytics pipelines, Section V summarizes three industrial case studies in which the pipeline approach has been applied. Section VI discusses the lessons learned and limitations. Section VII concludes this paper and discusses our future work.

II. BACKGROUND AND RELATED WORK

Three areas form the foundation for our contributions to industrial analytics: data science, big data architecture, and software quality attributes.

A. Data Science

The mission of data science [6] is to extract knowledge from data, with or without subject matter expertise, using scientific...
discipline. The steps are to collect and clean raw data, explore the characteristics and relationships, and develop models and algorithms that uncover patterns and predict outcomes. Finally, the results need to be delivered in a format and terms that can be understood by non-technical stakeholders.

The complexity of data science computations and the corresponding value of results evolves as more insight is discovered from the application domain. For example, as shown in Figure 1, historical data lends hindsight based on data exploration and correlation with influencing factors, followed by application of investigative algorithms motivating collection of additional data to enhance forecasting, and finally development of models providing better understanding of the domain, automating and optimizing subsequent calculations.

The most effective data scientists are typically well-versed in both science and technology. Given this background, there are no limits to solution alternatives. For example, algorithms and models can be constructed and executed in highly distributed deployments. This results in complex IT and software architectures specifically tuned to the application domains, and algorithms represented as coded programs: difficult to integrate and extend. In reality these cross-functional skills are difficult to find. For industrial analytics, business driven, not exploratory analytics is desired and unfortunately business goals do not translate directly into analytics projects.

B. Big Data Architecture

Standardization of big data architectures has accelerated with innovations in web search, especially using Apache Hadoop [8]. One key limitation of the original Hadoop MapReduce [9] framework is a restricted computational model. Subsequent to its emergence as the defacto large-scale data processing platform, the big data community recognized and advocated for a more generic and streamlined architecture platform with the realization that big data is not a single analysis pattern or a single data storage technology.

Hadoop vendors provide integrated platforms that support alternative processing engines like SQL, interactive and real-time queries, and streaming as part of their offerings. Subsequently, YARN [10] was introduced and adopted as a formal abstraction to encapsulate the need for resource management among different applications executing within a Hadoop cluster.

In a similar fashion to Hadoop and its distributions, the University of Berkeley's AMPLab [11] identified an alternate architecture for big data. From its inception, AMPLab introduced the Berkeley Data Analytics Stack (BDAS) with Spark [12] (in-memory computation engine alternative to disk-based Hadoop) at its foundation. BDAS aims to integrate batch analytics, streaming, SQL processing, graph analytics, and machine learning within the same framework. There are other platforms but Hadoop and Spark are the most common and popular big data architecture platforms.

C. Software Quality Attributes

Given the variety of choices for realizing analytics, it is difficult and time consuming for highly specialized domain experts to choose the optimal approach that creates business value. In software engineering, quality attributes of a software system refer to constraints how the system implements and delivers its functionality [14]. From a requirements engineering perspective, functional requirements define operations that the system must be able to perform. Non-functional requirements (NFR) describe how well the system must perform its functions, and how to measure these aspects of the system. Quality attributes are characteristics of NFRs, such as performance, reliability, scalability, security, and usability.

![Figure 1. Data science stages [7]](Image)

Improving one quality attribute of the system can have negative effects on other quality attributes. For example, making a system more secure can make it harder to use, or making it easier to use can make it less secure (security vs. usability); Using platform-specific features can make a system run faster, but that often makes it much more costly to port to another platform (performance vs. portability). Trade-offs must be made by taking into consideration the relative importance of each of the system qualities. Many of the important properties of analytics alternatives are quality attributes. This suggests the potential for a systematic approach for choosing the appropriate combination of analytics tools and algorithms for an application domain.

III. PIPELINE ARCHITECTURE

Successful analytics applications typically address a single problem and use specific technology choices. The next application needs to start from scratch or the previous solution is reused (copied), even if the trade-offs are ill-suited to the new problem. In this paper, we envision a product line approach [15] for analytics applications, by facilitating systematic reuse both on the infrastructure and software level, reducing the effort required to deliver analytic applications. A core platform provides the common functionality and infrastructure for industrial analytics applications, expediting application delivery. This lowers the knowledge and effort necessary to create business value from machine data.

Our core platform needs to be flexible enough to deliver different quality attribute trade-offs when it comes to the choices for industrial analytics applications. We therefore came up with the notion of an analytics pipeline; a flexible way of performing analytics with our core platform. A pipeline is an environment that allows specific technology choices that can store data and execute analytics programs. Analytics pipelines are composable in serial and parallel combinations.
Inspiration for our analytics pipeline comes from the work of Nathan Marz, who described the so-called Lambda Architecture (LA) pattern [16]. This pattern combines batch and speed (e.g. streaming) pipelines, called layers in the pattern, which work in parallel. All data coming to the system is dispatched to both pipelines. The batch pipeline is responsible for maintaining the master data set in an immutable way and to pre-compute the batch views. Another pipeline, the serving layer, takes the batch views and indexes them, so they can be queried in an ad-hoc and low-latency way.

The LA pattern has benefits and disadvantages. It offers a highly scalable and fast response system, which can reliably store and retrieve huge amounts of data. On the other hand, the code in the batch/serving and speed pipelines needs to be kept in sync for the system to deliver consistent results from the two data paths. Furthermore, operation costs are higher as different computing clusters have to be maintained for each pipeline.

This combination of data paths is a realization of a more generic architecture pattern in which different pipelines are composed to deliver necessary architectural qualities. Conceptually there is no reason why other pipelines could not be chosen and composed to get other trade-offs. For our core platform a one-size-fits-all approach does not meet our goals. Instead, flexibility in the platform to choose a trade-off that makes sense in the specific context in which the platform is required. We provide some examples of these specific trade-offs and contexts in Section V.

The reference architecture pattern that realizes this flexibility in our core platform is depicted in Figure 2. New data arrives at a Dispatcher, which sends the data to the relevant Analytics pipelines that supply an environment in which data is analyzed and stored. Clients can either pull data from the Analytics pipeline (e.g. by a query) or get data pushed to them (e.g. by being sent a notification). An Analytics pipeline can also feed the Dispatcher with data, composing with other pipelines that tap into these data streams. One analytics pipeline can feed an arbitrary number of other pipelines. Abstractly seen, our pattern is a more detailed and distributed instantiation of the pipes and filters [17] architectural pattern.

Figure 3 shows the components and connectors in an Analytics pipeline. First, the Ingress component is connected to the Dispatcher and has the responsibility to transform the incoming data stream into something the Storage* component can handle. Storage* is a (temporary or long term) resource (e.g. buffer, memory, disk, storage cluster, distributed file system), which makes the data available to other components. The Analytics component is configured with the algorithmic functionality, which is designed using the Engineering interface. A Scheduler manages concurrent access to storage and schedules the analytics tasks. Clients use the Client interface to access the data (including analytics results). The Outgress component is responsible for exposing the pipeline results.

Ideally, the core platform acts as Analytics as a Service (AaaS), similar to Platform as a Service (PaaS), to deploy analytics applications. The platform manages deployment of an application without the developer having to care about infrastructure, and decides and optimizes where and how the application is deployed.

For AaaS to work, an application must define additional meta-data. An ingress specification is needed to know the compatible pipeline types and which data the application depends on. A QoS specification defines the application runtime qualities (e.g. result delivery deadlines). An outgress specification defines the data exposed to other applications to make the pipeline composable.

To realize the analytics pipeline architecture pattern, many different alternatives can be used for the analytics technology choice and algorithm development style. See Table I below for a catalog of our selections.

IV. NON-FUNCTIONAL PROPERTY RATING OF THE PIPELINES

A group of architects was introduced to the analytics pipeline pattern and asked to rate the alternative choices. In this section, we present our approach for choosing the optimal analytics pipelines for an application.

A. Pipeline Identification and Rating Methodology

The ratings evaluation parameters for the pipeline types evolved over time, starting with simple computer resource requirements and later extending to software quality attributes. Based on the system context and capabilities of the core platform, we established a baseline by:

1. Identifying common pipeline types and general non-functional properties (such as performance and flexibility),
2. Conducting an internal individual survey among six architects for rating (scale 1-10) the different pipeline characteristics regarding these non-functional properties, and
3. Consolidating initial results and revising ratings in group meetings.

Outliers in the votes were identified based on the standard deviation of the ratings, where deviations greater than 2.0 were explicitly discussed to avoid misunderstandings. The architects revised their ratings after active brainstorming. Then the group iteratively expanded and refined the analytics pipelines, and clarified and refined the non-functional properties for subsequent votes.
Table II shows the final version of the non-functional properties with descriptions and definition of their scales. Each of the participants voted for the non-functional properties of the individual pipelines, where 1 means the corresponding non-functional property is low/bad and 10 means it is high/good. Figure 4 shows the rating dimensions and values for a selected set of analytics pipelines.

B. Pipeline Rating Results

Figure 5 shows the average rating properties compared for each pipeline type, and Figure 6 presents the pipeline rating profiles. Table III shows the averages and standard deviations based on the participant votes, where values in bold warrant further attention. Table IV summarizes our interesting findings based on the rating results.

C. Threats to Validity

There are several threats to the validity of the pipeline rating results. First of all, we may have missed non-functional properties that would have been important to make sound decisions for specific problems faced in practice. Additionally, the ratings we gathered were not supported by detailed benchmarks for the various non-functional properties but more on subjective experience by the people responding to the rating survey.

Furthermore, there are threats resulting from the execution of the rating itself. First, we had a small set of participants which all, despite a quite different background, were from the same company, namely ABB Corporate Research. Additionally, even though all of the participants have a strong technical background
in computer science, software architecture and data analytics, not all of the analytics pipeline technologies were known to the participants to the same extent. However, we argue that having the results based on a few experts available is already a good starting point and will improve when the rating process is executed for a broader audience.

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Data Flexibility</th>
<th>Algorithm Flexibility</th>
<th>Productivity</th>
<th>Static Capacity</th>
<th>Dynamic Capacity</th>
<th>Analytics Latency</th>
<th>Round Trip Response</th>
<th>Scalability</th>
<th>Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadoop</td>
<td>9.5</td>
<td>6.2</td>
<td>7.3</td>
<td>10.0</td>
<td>8.2</td>
<td>3.5</td>
<td>3.8</td>
<td>9.0</td>
<td>9.2</td>
</tr>
<tr>
<td>Indexed</td>
<td>8.7</td>
<td>4.8</td>
<td>7.3</td>
<td>9.7</td>
<td>8.2</td>
<td>6.2</td>
<td>5.0</td>
<td>9.0</td>
<td>8.7</td>
</tr>
<tr>
<td>RDBMS</td>
<td>7.8</td>
<td>5.7</td>
<td>8.8</td>
<td>5.3</td>
<td>5.2</td>
<td>5.0</td>
<td>4.3</td>
<td>4.3</td>
<td>5.0</td>
</tr>
<tr>
<td>Key-Value Pair</td>
<td>8.7</td>
<td>4.8</td>
<td>7.2</td>
<td>9.3</td>
<td>9.0</td>
<td>7.3</td>
<td>6.0</td>
<td>9.2</td>
<td>9.5</td>
</tr>
<tr>
<td>Time Series</td>
<td>4.5</td>
<td>3.2</td>
<td>7.8</td>
<td>9.2</td>
<td>9.0</td>
<td>7.3</td>
<td>6.2</td>
<td>9.2</td>
<td>9.3</td>
</tr>
<tr>
<td>Streaming</td>
<td>9.3</td>
<td>7.8</td>
<td>5.3</td>
<td>1.5</td>
<td>8.7</td>
<td>9.2</td>
<td>8.8</td>
<td>8.7</td>
<td>6.5</td>
</tr>
<tr>
<td>In-Memory</td>
<td>9.2</td>
<td>8.5</td>
<td>7.8</td>
<td>9.3</td>
<td>5.8</td>
<td>8.0</td>
<td>3.7</td>
<td>8.0</td>
<td>6.3</td>
</tr>
<tr>
<td>Single Node</td>
<td>7.2</td>
<td>9.2</td>
<td>8.2</td>
<td>3.0</td>
<td>2.4</td>
<td>3.4</td>
<td>3.2</td>
<td>2.0</td>
<td>2.6</td>
</tr>
<tr>
<td>Graph</td>
<td>6.8</td>
<td>6.3</td>
<td>6.5</td>
<td>9.2</td>
<td>7.8</td>
<td>8.2</td>
<td>5.0</td>
<td>9.0</td>
<td>7.8</td>
</tr>
<tr>
<td>Custom</td>
<td>9.5</td>
<td>9.5</td>
<td>3.0</td>
<td>5.5</td>
<td>5.7</td>
<td>5.2</td>
<td>5.2</td>
<td>4.7</td>
<td>4.5</td>
</tr>
</tbody>
</table>

V. APPLICATION CASE STUDIES

In this section, we present three industrial analytics cases that use some of the analytics pipelines explained above. The general description of each case is presented as well as the characteristics of the data they generate. Based on the data and the expected analytic functionality, the main quality attributes of each case are identified, which provide the rationale behind the selection of analytics pipelines for each case. The ideal rating values for each application case are shown in Figure 7, overlaid with the selected analytics pipeline ratings for comparison.

A. Case 1: Wide-Area Monitoring, Protection and Control of Electric Power Systems

Interconnections for electric power production, transmission and distribution make up one of the most complex systems created by humans. Large systems like the North American or the European interconnections typically consist of thousands of generators, transmission lines, substations and electricity consumers. Power systems need to be constantly operated such that they remain at a dynamic equilibrium, despite the uncontrollable, nature of some of their components (electric loads, renewable generation), and stability needs to be maintained in face of unexpected events, like the loss of a transformer, a transmission line or a generating unit.

Transmission system operators install measurement devices and software systems, that allow them to monitor in real-time the dynamic state of their networks, so that proper stabilization actions are timely taken when needed. These measurements are communicated from their respective remote locations to a control center where algorithms are executed in real-time. The algorithms take this time series data and perform functions such as system state estimation, identification and monitoring of electromechanical oscillations, calculation of distance from voltage instability and others using streaming analytics pipelines.

Protection schemes are designed to allow for automatic actions that stabilize the system when needed: opening switches, modifying setpoints of power flow control devices, changing generator setpoints and others. Such wide-area monitoring, protection and control systems have strict time and reliability requirements; the stabilizing action must be at the right moment and in the right location, properties that fit with a streaming analytics pipeline for detecting events and coordinating the appropriate actions in real-time.

The most important quality attributes for this application include: static capacity, analytics latency, round-trip response, scalability and reliability. To this purpose, the streaming and time-series analytics pipelines are the most suitable. Figure 7(a) shows the streaming, time series, and in-memory analytics pipelines used and the ideal application ratings outline.

B. Case 2: Fault Event Prediction in Power Distribution

Distribution systems take power from bulk-power substations to consumers. Power substations and their associated equipment play an essential role in the distribution of electricity. Much of the power distribution infrastructure in the western world is over 50 years old. A key issue facing utilities is to leverage their limited funds for maintenance and repair of power distribution lines. Studies in the UK show that more than 70% of unplanned loss of electrical power is due to problems in the distribution grid [18]. According to a survey conducted by the Lawrence Berkeley National Laboratory, power outages or interruptions cost the United States of America $80 billion annually [18].

It is desirable to have automated prediction models that can forecast when a fault event may occur in a distribution network. Moreover, the efficiency of dispatching crews can be improved by having automated diagnostic methods and fault predictive capabilities. The analyses conducted to predict faults in the power distribution grid use historical data from weather conditions, grid electric value readings at the time of a fault event, power loadings, and the type of distribution grid infrastructure. The analysis needs to be in near real-time.

To achieve the desired analytic results, a streaming analytics pipeline implements this application as latency, round-trip response, scalability, data flexibility, and algorithm flexibility are the most important quality attributes. In this application, the requirements for data flexibility, algorithm flexibility, analytics latency, round trip response, and scalability are quite high due to the near real-time requirements. Figure 7(b) shows the qualities of the streaming analytics pipeline used and the ideal application ratings outline.
TABLE IV. PIPELINE RATING FINDINGS

<table>
<thead>
<tr>
<th>Type</th>
<th>Finding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadoop</td>
<td>Top rated in both data flexibility as well as static capacity.</td>
</tr>
<tr>
<td>Indexed</td>
<td>Balanced overall rating</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Mediocre, except for availability of good tools and experts for algorithm development</td>
</tr>
<tr>
<td>Key-Value Pair</td>
<td>High dynamic capacity, good scalability and the best reliability.</td>
</tr>
<tr>
<td>Time Series</td>
<td>Similar to key-value pair with reduced flexibility</td>
</tr>
<tr>
<td>Streaming</td>
<td>Best analytics latency and round-trip response time</td>
</tr>
<tr>
<td>In-Memory</td>
<td>Balanced overall rating</td>
</tr>
<tr>
<td>Single Node</td>
<td>Least attractive properties except for flexibility and productivity.</td>
</tr>
<tr>
<td>Graph</td>
<td>Balanced overall rating</td>
</tr>
<tr>
<td>Custom</td>
<td>Best flexibility but worst in productivity.</td>
</tr>
</tbody>
</table>

Figure 5. Ratings for the different pipelines in comparison with highlighting of top-rated pipelines

Modern control systems produce large quantities of data and potentially can show a large volume of alarms to the operator. Most often the major usability problem is too many alarms annunciated in a plant upset, commonly referred to as alarm flood or alarm burst. There can also be other problems with an alarm system such as poorly designed alarms, improperly set alarm points, ineffective annunciation, unclear alarm messages, and others.

Industrial plants operate more efficiently with capabilities that dynamically filter the process alarms based on historical plant operation and conditions so that only the significant alarms are annunciated to the operators. Learning from large volumes of historical alarm data helps reduce the number of redundant and nuisance alarms, predict alarms given certain events, and identify tripping alarms that may cause costly stoppages.

To achieve the required analytic results in this application, streaming and graph analytics pipelines were used. The most important quality attributes in the application include: algorithm flexibility, scalability, productivity, and reliability. The requirements for algorithm flexibility and scalability are quite high as different analytic methods have been developed to analyze alarms. Figure 7(c) shows the streaming and graph analytics pipelines used, and the ideal application ratings outline.

VI. DISCUSSION

The evolution of our pipeline pattern reference architecture and analytics pipeline types has its roots in our collaborative agile architecture process [19]. In this section we summarize the lessons learned and limitations of our investigation.
A. Lessons Learned

The technology choices for analytics alternatives are rapidly evolving. Basing our reference architecture on the current (or even proposed future) industry solutions is risky at best. The architectural decisions need to be driven by business and common customer needs with the potential for generating business value. Our reference architecture is described in terms of architecture concerns and the forces that drive business needs.

Very few industrial organizations have the resources and luxury of extended time to market to develop custom analytics infrastructures. Leveraging open source community work products can leap business initiatives forward. With that said, combining disparate release versions to create best of breed deployments is fraught with pain. It is more efficient to start with a commercial analytics framework distribution where those issues have already been resolved.

Systematic pipeline type selection is enabled by our evaluation process and over time each pipeline type becomes easier to deploy. The experience gained from creating an implementation is captured so the next implementers do not start from scratch. Validation of the business value is added to community knowledge to improve confidence in the specific pipeline ratings.

B. Limitations

Our reference architecture pattern is not an optimal solution and may not even be feasible to realize. It is a trade-off between flexibility and complexity: complexity in IT operations as well as the lack of data transparency. Each pipeline type brings with it a set of algorithm development styles that may not be compatible with the engineering staff holding essential subject matter expertise for the application domain.

The pipeline rating process surveys technology experts for their guidance on the value of each pipeline property. An expert with affinity to a specific pipeline type is hopeful that all properties are maximized, based possibly on the future potential of the technology stack as opposed to its current practice. The ratings are therefore driven by subjective perceptions and would benefit from a significantly larger audience for consistency.

VII. CONCLUSIONS AND FUTURE WORK

Industrial analytics provides a framework to derive actionable knowledge for critical decision making from data generated and collected from industrial systems, machines, equipment, and industrial processes. The primary raw material in industrial analytics is machine data. Readings generated in industrial systems can originate from different sources, be of various forms, and arrive at large rate ranges. Depending on the types of decisions made in industrial systems, analyses require responses with a spectrum of time intervals.

Throughout our experiences in developing architectures for a variety of industrial analytics applications, the concept of analytics pipelines has played a key role to define the “right” architecture. Analytics pipelines provide a framework for a product line environment to develop industrial analytics applications. The analytics pipeline approach defines a reference platform that delivers an infrastructure to develop concrete architectures satisfying their quality attributes.

The core platform described in this paper has proven to be very flexible in providing different quality trade-offs for architecting industrial analytics applications. An analytics pipeline provides an environment with specific technology choices to store data and execute analytics that result in actionable knowledge.

The work presented in this paper will continue to evolve as new applications are developed. For example, new quality attributes such as security, availability, maintainability, will need to be formalized and incorporated into the current analytics pipelines structure in the future. Moreover, the analytics technology continues to proceed at very rapid pace and this evolution is expected to bring additional dimensions that will need to be considered in the analytics pipelines platform.

REFERENCES