Available online at www.sciencedirect.com

ScienceDirect

IFAC-PapersOnLine 49-16 (2016) 330–335

Cloud-based Decision Support and Automation for Precision Agriculture in Orchards

Li Tan
School of Electrical Engineering and Computer Science and Center for Precision & Automated Agricultural Systems, Washington State University

Abstract:
Recent technological and commercial developments make cloud computing an affordable, scalable, and highly-available platform technology. Meanwhile, precision agriculture is showing its potentials by improving agricultural operations through better data-driven decision making. Nevertheless, further development of precision agriculture requires better technology and tools to process data efficiently at a reasonable cost, and to translate the data to better decisions and actions in a field. We developed a framework for cloud-based Decision Support and Automation systems that can acquire data from various sources, synthesize application-specific decisions, and control field devices from the Cloud. A distinctive feature of our framework is its extensible software architecture: decision modules can be added and/or configured for a specific operation. The platform features a device-agnostic frontend that can process incoming data in different formats and semantics. Finally, the platform incorporates software-defined control, a new software design paradigm we proposed to enable versatile and safe control of field devices from a cloud computing platform. An early version of the system has been developed and tested with support from the USDA.

Key words: cloud computing, decision support and automation, precision agriculture, software-defined control.

1. Introduction

Precision agriculture is a site-specific farming practice that uses technologies to measure and respond to inter and intra-field variability in crops. It has been seen as a critical tool for increasing agricultural productivity while preserving natural resources. The “brain” of precision agriculture is a decision support system (DSS) that helps a grower process and respond to intra- and inter-field data. With recent technological advances, particularly sensors technology and online data services, agriculture is increasingly becoming a data-rich operation. Particularly specialty crop industry in US is going through a transformation through the use of information technology. By the USDA definition, specialty crops are “fruits and vegetables, tree nuts, dried fruits, horticulture, and nursery crops (including floriculture).” Specialty crops are responsible for half of the gate value of US agricultural output [1]. Compared with commodity crops, special crops are more site-specific and labor-intensive. It is sensitive to variation in field, labor situation, and many other factors. Whereas many agriculture operations can use a decision support system, specialty crops particularly may benefit from a better decision support system, as a majority of orchards today still follow a traditional human-centric decision process.

Developing a better decision support system for specialty-crop industry presents some specific challenges in information technology. Specialty-crop operations are highly seasonal. Requests for decision support can fluctuate drastically on- and off-seasons. One challenge is *how to meet the fluctuating demands while still providing highly available services*. Moreover, each orchard has its own unique operation characteristic. Another challenge is *how to develop a decision support system that can be reconfigured for a specific operation*. Finally, a precision agriculture operation is a close-loop control system, with inputs (e.g. sensors and other data sources) from a field and feedbacks (e.g. field actions) to the field. A traditional decision support system emphasizes only on the first part of the control loop, that is, taking inputs and synthesizing decisions. A grand challenge is *how to close the control loop by controlling field devices safely with optimized decisions.*

According to the official NIST definition, cloud computing is "*a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.*"[2]. Cloud computing is inspired by a utility delivery

10.1016/j.ifacol.2016.10.061

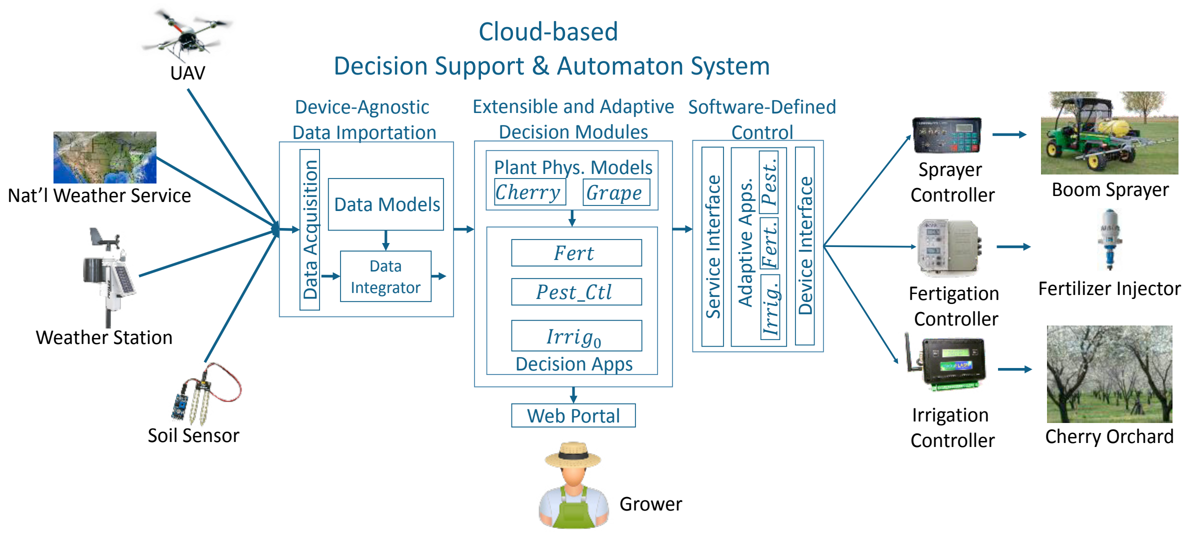
model: computing powers may be delivered on demand just like electricity or gas, in an effort to meet demands for scalable, highly-available, and economical Internet-connected computing resources. A cloud computing service enables users to request and release resources on demand, and to pay for only what has been used. By pooling resources together, a cloud computing service generally has a higher availability than traditional user-operated server networks.

Cloud computing is particularly beneficial for decision support in precision agriculture for specialty crops. First, precision agriculture in an orchard is a data-rich operation. A decision support system needs to handle a large volume of data from sensors and other sources. Cloud computing provides scalability necessary for handling these data in real time. Second, the demand for decision support fluctuates greatly on- and off-seasons. Through resource provisioning [3], cloud computing can change quickly the number of server instances and other resources, based on the demand. Finally, agriculture decision support systems are increasingly hosted on Internet, to take advantage of internet-connected devices (Internet of things[4]) and to build an online community. A cloud service provider handles the complexity of running hardware and maintaining middleware for enhanced availability and security. This leaves developers to focusing on the logics of a web-based decision support system.

Moving towards cloud-based decision support presents opportunities as well as new challenges. First, precision agriculture uses a variety of sensors and data sources, each of which may have its own data format and semantics. A cloud-based decision support system needs to handle a diversified profile of data types and formats; second, traditionally a decision support system is application-specific. A farmer may need to access different systems for a specific application (e.g. irrigation, fertilization, etc). To provide a streamlined user experience, a cloud-based decision support system shall be able to be extended and configured for different applications; and finally, recent development of Internet of Things (IoT) links field devices through Internet. To capitalize the progress in IoT, a future decision support system is expected to control field device safely from the cloud.

In this paper we discuss our framework for cloud-based decision support and automation systems (DSAS), and our experience of implementing it in Agrilaxy, a DSAS we developed from ground up to take advantage of the scalability and availability of a cloud computing platform. We developed new techniques to address the design challenges faced by a cloud-based DSAS. The rest of the paper will be organized as follows: Section 2 gives an overview of our framework. In Section 3 we discuss its device-agnostic data importation frontend, which works with different data sources with custom-defined data format and semantics. In Section 4 we discuss its extensible software architecture for decision modules. In Section 5 we introduce *Software-Defined Control* paradigm, a new software design paradigm for controlling physical devices in a field from the cloud. Section 6 discusses the current implementation of Agrilaxy, Finally, Section 7 concludes this paper.

Figure 1 Cloud-based decision support & automation system: an overview



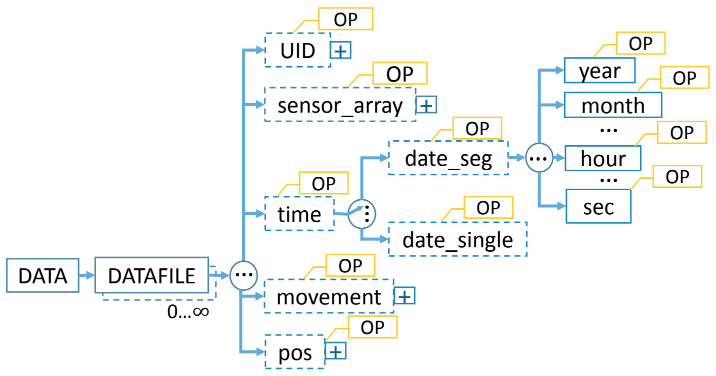
2. System overview

Figure 1 shows the overview of our framework for cloud-based DSASs. It has a device-agnostic data importation frontend. The front end uses data models to define the formats and semantics of input data sets. The details of a data model are given in Section 2. Its decision modules are developed with an extensible software architecture, featuring a hierarchical modular design. A module, referred to as a web app in the DSAS, is characterized by its interfaces. A new web app can be added to decision module hierarchy on-the-fly, or used to replace an existing module with a compatible interface. The details of our extensible software architecture are introduced in Section 3. In Section 4, we introduce Software-Defined Control, a new software design paradigm for managing the complexity of controlling a diversified profile of devices. Software-defined control virtualizes physical devices through layers of abstraction. Each layer abstracts the implementation details of the layer underneath, while providing a uniform control interface to the layer above.

3. Device-agnostic data importation

Precision agriculture is characterized by the use of different sensors (e.g. soil moisture sensor, nitrogen sensor) and data sources (e.g. weather site) to measure intra- and inter-field variability. A cloud-based DSAS needs to work with a wide variety of sensors and data sources. To handle the diversity of data from different sources, we extend a meta-data-model-based technique initially proposed in [5]. The technique uses a data model to specify the format and data types of an input data set, e.g., temperature, soil moisture reading, etc. A meta data model is a model of data models. It specifies permissible data types for columns, and operation semantics associated with each data type. Operational semantics for a specific data type specifies operations that can be performed on the data type. For example, an operational semantic for temperature in Fahrenheit may define how to translate data to an intermediate format in Celsius. To improve inter-operability, we specify a data model in XML [6], which is a prevailing data format used to exchange structured data over Internet. A meta data model is specified in an extension of XML schema (XSD) [7], referred to as XSD*. XSD* extends XSD with operational semantics for a data type. Figure 2 illustrates a meta-data model in the proposed XSD*. Elements of the model are extended with their operational semantics (i.e. 'op'). For example, element 'time' has two children 'date_seg' and 'date_single', each representing a possible format of time. A time in the 'data_seg' further contains 'year', 'month', etc. The 'op' associated with 'date_seq' specifies an algorithm translating a time in 'date_seq' to a unified internal representation accepted by decision modules. For each data model, a data importer is generated to handle data sets defined by the data model. The data importer will execute the operational semantics defined in the meta-data model, on an input data set. This will translate the data set into an intermediate format accepted by decision modules. To introduce a new type of data source (e.g. a new type of sensor), one only needs to define its data model in XML. The DSAS will translate the XML data model to a data importer.

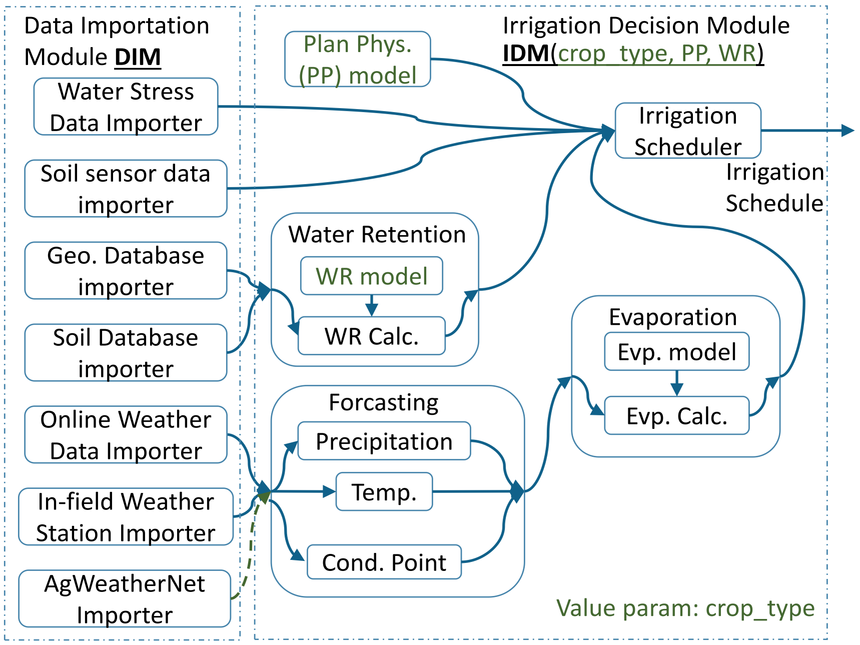
*Figure 2: Meta data model in XSD**



4. An extensive software architecture for decision modules

At core of a decision support system is its decision logic. In a traditional decision support system the decision logic is hard-coded for a specific application. Nevertheless, each orchard operation is different, and it is affected by many factors such as the orchard's crop types, environment, and even business model. We proposed an extensible and adaptive software architecture that enables a DSAS to be easily extended with new decision logics. The software architecture follows the open/close principle, an important principle in object-oriented software design, i.e., a software shall be open for extension but close for modification [8]. In the context of decision modules, it means that a system can be easily extended with new decision modules with minimal change to its implementation. Following this principle, we developed an extensible software architecture for building highly customizable decision modules. Figure 3 shows a variety of reconfiguration mechanisms supported by the extensible software architecture, including dataflow configuration, and value and block parameters.

Figure 3 Extensible dataflow-driven software architecture



The extensible software architecture uses a data-flow-driven design, giving a user an intuitive way to view and configure a decision logic. In the example shown in Figure 3, a user may re-route block 'Forcasting' to take inputs from the AgWeatherNet importer instead of the in-field weather station importer. This enables the decision module to use weather data from AgWeatherNet, a network of weather stations in WA [9]. The data-flow-driven design also enables behavior and structural hierarchies (as shown in Figure 3). To reduce the overhead of building a new decision module from scratch, the software architecture allows a user to replace only a portion of the logic as necessary, through value and block parameters. For example, in Figure 3 parameter 'crop_type' is used to pass the crop type to the decision module and it is passed down in the block hierarchy using a scoping rule.

5. Software-defined control: a software design paradigm for cloud-based control

Compared with existing DSSs, a distinctive feature of our cloud-based DSAS is its ability to control field devices directly from the Cloud. Growers often deploy a variety of devices for different applications (e.g., irrigation, pest control). These devices come with their own proprietary control interfaces. A challenge in cloud-based control automation is *how to control a variety of field devices from the Cloud*. To address this challenge, we proposed *software-defined control*. Inspired by the concept of software-defined network[10], software-defined control makes devices interoperable by wrapping them with a unified interface defined by software. This will give a device maker a flexible (interface is defined by software, not hardware) and non-intrusive (no modification of their device interfaces is necessary) way to make their devices work with a DSAS. Software-defined control enables multi-stage software-enabled adaptation from device-independent decisions to device-specific control signals. By separating design concerns, software-defined control structures the adaptation in a three-layer architecture: 1) a decision plane synthesizing device-independent decisions. These decisions specify required actions at a high level, e.g., a geographic distribution of irrigation volume; 2) a service plane translating a decision to schedules for each device. A schedule reflects the configuration and layout of devices; and 3) the device plane executes the schedules. Figure 4 illustrates software-defined control, using a 3-zone irrigation system as an example. Each irrigation zone has its own controller (Dev_{irri0}, Dev_{irri1}, and Dev_{irri2}, respectively) controlling a set of valves. A decision made by the decision plane specifies a distribution of water volume in an orchard. The service plane translates the decision to irrigation schedules for each controller, and finally the device plane, consisting of three controllers, use the schedules to control valves.

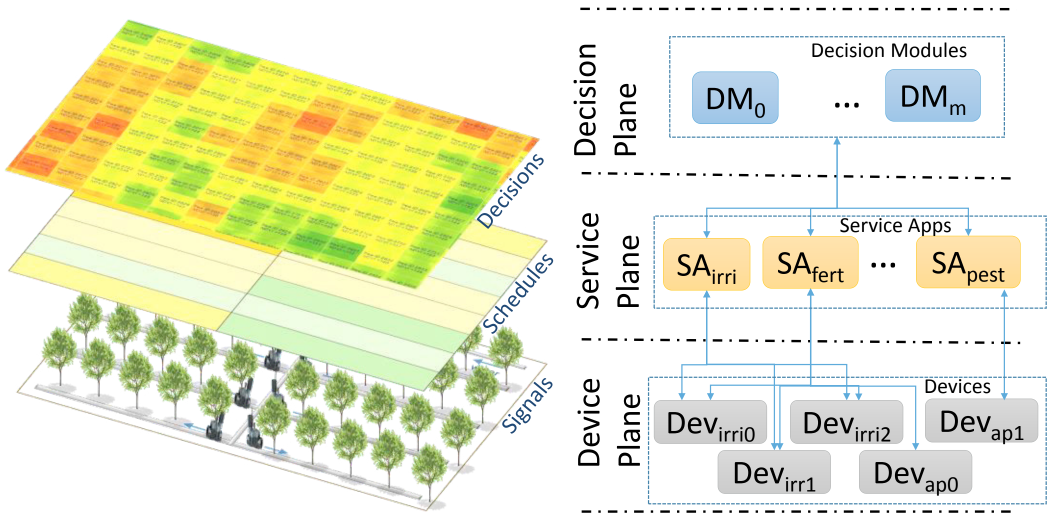


Figure 4 Software-Defined Control: An Overview

Layers of abstraction deployed by software-defined control enable each layer to provide an abstract view of the layer beneath through a well-defined interface. For instance, a service plane has a collection of service applications (SAs). Each SA provides an abstraction of a service provided by field devices. In the example in Figure 4, SA_{irri} provides an abstract view of an irrigation system, abstracting away details such as zoning and device types. This abstraction separates decision intelligence from devices and enables the development of device-independent decision modules.

Software-defined control also enables constant feedback and monitoring at each layer of abstraction. For example, the interface of a service plane allows a decision plane to discover SAs in the service plane, to send decisions to the SAs, and also to receive runtime statistics from them. The interface of a device plane enables two-way traffic in which the service plane can program field devices, and receive feedback from these devices. This interface allows the decision plane to use feedback from devices to adapt its own logics through its adaptive architecture.

6. System implementation

The initial version of Agrilaxy has been implemented in May 2014, which implemented device-agnostic data acquisition front-end. The system was developed using Ruby. Ruby is a dynamic-type programming language popular with web developers. Part of its popularity with web community is due to Rails, a ruby library framework developed specifically for server-side web development[11]. We tested Agrilaxy on Amazon's cloud-based service AWS [12]. Currently we are working on a new version of Agrilaxy, with emphases on implementing a cloud-based control automation backend and adaptive decision modules.

7. Conclusions

Cloud computing is becoming a mainstream choice for hosting web-based information systems. Cloud computing provides a scalable and highly available computing platform with on-demand resource allocation. It helps address challenges faced by a web-based agricultural decision support system. To take advantage of cloud computing, as well as to close the control loop in precision agriculture, we propose a new design framework for cloud-based Decision Support and Automation systems. The framework-features a device-agnostic data importation frontend, an extensible software architecture for decision modules, and cloud-based control automation using software-design control. The device-agnostic frontend supports input data set from different data sources; the extensible software architecture enables a DSAS to be extended for different decision applications; cloud-based automation closes the decision loop, from sensor inputs, to decision making, to control signals to a diversified profile of field devices. We implemented an initial version of our framework using Ruby on Rails, and tested it on Amazon cloud services (AWS).

References

[1] United States Department of Agriculture, “Specialty Crop Research Initiative.” [Online]. Available: http://nifa.usda.gov/sites/default/files/resources/SCRI Self-Study document.pdf. [Accessed: 14-Jun-2015].

[2] P. Mell and T. Grance, “The NIST definition of cloud computing,” 2011.

[3] C. Li and L. Y. Li, “Optimal Resource Provisioning for Cloud Computing Environment,” *J. Supercomput.*, vol. 62, no. 2, pp. 989–1022, 2012.

[4] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Futur. Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[5] L. Tan, R. Haley, and R. Wortman, “An Extensible and Integrated Software Architecture for Data Analysis and Visualization in Precision Agriculture,” in *the proceedings of IEEE Internation Conference on Information Reuse and Integration (IRI’09)*, 2009.

[6] W3C Working Group, “RDF 1.1 XML Syntax,” *W3C Recommendation*. 2014.

[7] W. C. W. D. December, “W3C XML Schema Definition Language (XSD),” *Language (Baltim).*, 2009.

[8] B. Meyer, “Tell less, say more: the power of implicitness,” *Computer (Long. Beach. Calif).*, vol. 31, no. 7, pp. 97–98, Jul. 1998.

[9] Washington State University, “AgWeatherNet,” 2015. [Online]. Available: http://weather.wsu.edu/awn.php.

[10] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-Defined Networking: A Comprehensive Survey,” *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[11] “Ruby on Rails.” [Online]. Available: http://rubyonrails.org/. [Accessed: 15-Jul-2015].

[12] Amazon, “Amazon Web Services,” 2016. [Online]. Available: http://aws.amazon.com. [Accessed: 03-Jan-2016].