Hacia un marco de trabajo para desarrollar sistemas de programación de la producción utilizando servicios en la nube.

Daniel Díaz[1,2], Sandra Oviedo[1,2], Fernando Guardia[1,2], Juan Manuel Cuneo[2], Alejandra Otazú[2]

[1] Laboratorio de Informática Aplicada a la Innovación
Instituto de Informática, [2]Departamento de Informática
FCEFN- UNSJ

{ddiaz, soviedo, fguardia, jmcuneo, aotazu}@iinfo.unsj.edu.ar
San Juan, Argentina

Resumen. Desde la perspectiva de la Ingeniería Industrial un sistema de programación de la producción es el corazón de cualquier industria dedicada a la manufactura de cualquier producto. Este tiene como responsabilidad determinar que tarea deberán realizar determinados recursos productivos, tales como máquinas u operarios, en un determinado momento, para cumplir con los objetivos que impone la dirección de la empresa. La programación de la producción es parte del sistema de planificación y control de la producción, el cual es parte del sistema logístico de una empresa. Desde una perspectiva matemática, un sistema de programación de la producción trata con un conjunto de problemas de programación de la producción. En la búsqueda de soluciones a estos problemas intervienen las Ciencias de la Computación. A lo largo de tiempo los investigadores han desarrollado muchos algoritmos para resolver este tipo de problemas, pero lamentablemente gran parte de esta investigación no ha sido reflejada en el campo industrial, este problema se conoce como el gap entre la academia y la industria. Una forma de vincular academia con industria son los sistemas de programación de la producción, por lo tanto, proponer nuevas formas para construir sistemas de programación de la producción ayuda a reducir este gap. Este trabajo propone un marco para desarrollar sistemas de programación de la producción utilizando servicios en la nube.

Palabras clave: scheduling systems, microservicios, cloud services.

1 Introducción

Las herramientas informáticas que permiten solucionar problemas de optimización industriales se denominan "optimization solvers", son plataformas que permiten modelar, diseñar e implementar soluciones para problemas de optimización. Los problemas de scheduling son un caso particular de los problemas de optimización [1] y

en la industria se los denomina problemas de programación de la producción, de aquí en adelante cuando se utilice la palabra scheduling se estará haciendo referencia a este tipo de problemas. Un sistema de scheduling trata con muchos problemas de scheduling [2]. En una planta industrial, un sistema de scheduling es el corazón del sistema de planificación y control de la producción. Según la arquitectura jerárquica ANSI/ISA 95 que guía la organización de una planta industrial, el sistema de scheduling interactúa con otros sistemas de fabricación, tales como ERP (Entreprise Resource Planning) y MES (Manufacturing Execution System). En los últimos años, con el surgimientos de la industria 4.0 el proceso de planificación y programación de la producción está en constante cambio debido a que las funciones del MES han sido absorbidas por el CPPS (Cyber-physical production systems) [3], [4]. Existe una literatura muy extensa sobre scheduling y se ha escrito mucho sobre modelos y algoritmos que resuelven problemas de scheduling. Sin embargo, poco son los artículos que tratan de como traer estos modelos y algoritmos a implementaciones reales. Esto se conoce como el "hueco o gap" entre la teoría y la práctica de scheduling [5]. En [6] se propone al desarrollo de software como un camino para cerrar este hueco y se enfatiza en la necesidad de proponer nuevos caminos para diseñar e implementar sistemas de scheduling. Hoy en día, el desarrollo de software está en un constante cambio movilizado por las tecnologías de la computación en la nube. En el entorno industrial estas tecnologías están permitiendo que las pequeñas y medianas empresas utilicen las mismas tecnologías que las grandes empresas. Antes, este tipo de tecnologías solo eran accesibles a las grandes empresas debido a los costos que ellas implicaban. Los últimos avances tecnológicos han logrado que los solucionadores de optimización, orientados a resolver problemas de scheduling, se ofrezcan como servicios económicamente accesibles en la nube. Sin embargo para hacer un uso eficiente y eficaz de estos servicios se necesitan nuevos mecanismos que habiliten el desarrollo de sistemas de scheduling con estas tecnologías. Estos sistemas deben ser diseñados para aprovechar toda la potencia que brinda la nube en cuanto a escalabilidad, tolerancia a fallos, velocidad de cómputo, y economía de uso. En este artículo se propone un marco de trabajo para desarrollar sistemas de programación de la producción utilizando servicios en la nube, dicho marco está basado en el estilo arquitectónico de microservicios [7].

2 Evolución de los sistemas de scheduling

La investigación en el diseño y desarrollo de sistemas de scheduling se puede clasificar en 5 generaciones. Inicialmente, esta clase de problemas fue abordada por el área de la ingeniería industrial, y posteriormente por el dominio de la investigación operativa. En los comienzos del siglo pasado, Henry L. Gantt, fue uno de los primeros en proponer un sistema de scheduling: daily balance "el equilibrio diario" que consiste en un método de programar y de registrar el trabajo. En 1960 se desarrolló Planalog Control Board, una herramienta que constaba de una pizarra y entre otras cosas podía forzar restricciones de precedencia. Los primeros sistemas de scheduling computarizados nacieron con el MRP (Material Requeriment Planning). En la segunda generación surgen los ISS (Intelligent Scheduling Systems), tales como ISIS y OPIS de [8]. Esta fue una etapa donde el problema de scheduling se comenzó a abordar con técnicas de inteligencia artificial tales como algoritmos genéticos, simulated annealing, redes

neuronales y constraint satisfaction techniques. En esta generación aparecen los sistemas de scheduling que se construyen utilizando técnicas de KBS (Knowlegde Based Systems). La tercera generación se inició con el surgimiento del paradigma orientado a objetos, dando origen a sistemas de scheduling desarrollados con esta tecnología. Esto permitió el desarrollo de frameworks especializados en scheduling, los cuales encapsulan técnicas de inteligencia artificial [9]. La cuarta generación de sistemas de scheduling es la que se integra en lo que se conoce como Avanced Planning and Scheduling Systems [10]. La quinta generación está surgiendo actualmente y está basada en los sistemas ciberfísicos o CPS. En [11] y [12] se los define como sistemas de entidades que colaboran computacionalmente, que están en conexión intensiva con el mundo físico circundante y sus procesos en curso, proporcionando y utilizando, al mismo tiempo, servicios de acceso y procesamiento de datos disponibles en internet. En [12] se extiende el concepto básico de CPS a los sistemas de producción lo que da como resultado el concepto de Sistemas de Producción Ciberfísicos CPPS. En [13] se presenta una definición de Industria 4.0 que coloca a los CPS como uno de los elementos básicos que posibilitan el surgimiento de este paradigma. El proceso de scheduling de producción se ve notoriamente afectado por el surgimiento de la sistemas ciberfísicos, en [14] se discute una propuesta para tratar con los problemas de Job-shop Scheduling en el entorno de un sistema ciberfísico. En [15] se introduce un nuevo esquema de toma de decisión denominado Smart Scheduling capaz de hacer frente a los cambios propuestos por la industria 4.0 en lo que respecta a la toma de decisiones de un sistema de scheduling de producción. En [3] se discute como la arquitectura jerárquica ANSI/ISA 95 ha sido modificada por la industria 4.0 y más específicamente como estos cambios impactan en el proceso de scheduling de un industria.

3 Computación en la nube

Según [16] Cloud Computing es un modelo para habilitar acceso conveniente por demanda a un conjunto compartido de recursos computacionales configurables, por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios, que pueden ser rápidamente aprovisionados y liberados con un esfuerzo mínimo de administración o de interacción con el proveedor de servicios. Este modelo de nube promueve la disponibilidad y está compuesto por cinco características esenciales, tres modelos de servicio y cuatro modelos de despliegue.” Esta definición destaca que cloud es un modelo de prestación de servicios cuya principal orientación es la escalabilidad. Desde una perspectiva económica los servicios son elásticos, donde el usuario paga por lo que consume. Este modelo está compuesto por cinco características esenciales, tres modelos de servicio y cuatro modelos de despliegue. Las características esenciales son las que describen al modelo de cloud computing, en [17] se presentan como Auto-servicio por demanda, Acceso amplio desde la red, Conjunto de recursos, Rápida elasticidad, Servicio medido. Los Modelos de Servicios se refieren a la forma en la cual están organizados los servicios de cloud computing, son tres a saber, *IaaS* (Infrastructure as a Service), *PaaS* (Platform as a Service), *SaaS* (Software as a Service), una descripción detallada se puede ver en [18]. Los Modelos de Despliegue hacen referencia a cómo se

monta y pone en funcionamiento el sistema. Estos se presentan en [17] son *Nube privada, Nube comunitaria, Nube pública, Nube híbrida*.

4 Despliegue y Arquitecturas de aplicaciones en la nube

El desarrollo de aplicaciones basadas en los modelos IaaS o PaaS al momento del despliegue de las mismas se debe considerar los requerimientos sobre las capacidades de la computación en la nube que necesitan de las aplicaciones, tales como escalado automático, performance, despliegue en caliente, disponibilidad, monitoreo dinámico entre otros. Una arquitectura de una aplicación en la nube, entre otras cosas, está influenciada por las capacidades de la computación en la nube que se desean aprovechar. Generalmente, el desarrollo de aplicaciones en la nube es llevado a cabo ya sea siguiendo el estilo arquitectónico monolítico o siguiendo el estilo arquitectónico de microservicios [7] que se describen a continuación.

4.1 Arquitectura monolítica

Una aplicación para la nube basada en el estilo monolítico normalmente se compone de tres partes [7], en el lado del cliente está la interfaz de usuario, la cual normalmente se compone por un archivo HTML, un archivo javascript y una hoja de estilo. Estos archivos se ejecutan sobre un browser instalado en la máquina del usuario. El segundo componente es una base de datos, que contiene todos los datos con los que trabaja la aplicación. Por último, en el lado del servidor existe una aplicación que gestiona las solicitudes HTTP, ejecuta la lógica del dominio, lee, escribe y modifica datos de la base de datos, y construye los archivos HTML que son enviados al cliente. Esta aplicación es monolítica, todo está empaquetado como un único ejecutable. Cualquier cambio en el sistema implica construir y desplegar una nueva versión de la aplicación en el servidor. En [19] se enumeran una serie de inconvenientes de las aplicaciones monolíticas:

- Las aplicaciones monolíticas de gran tamaño son difíciles de mantener y evolucionar debido a su complejidad.
- La localización de errores requiere tiempo y un esfuerzo considerable de análisis del código.
- Sufren el "infierno de dependencia", una problemática que sucede cuando al agregar o actualizar bibliotecas da como resultado sistemas inconsistentes que no compilan / ejecutan o, peor aún, tienen mal comportamiento.
- Cualquier cambio en un módulo requiere reiniciar toda la aplicación. En proyectos de gran tamaño implica dejar a los usuarios sin sistema por tiempo considerables. Esto dificulta las pruebas y el mantenimiento del sistema.
- Dificultad para optimizar recursos de infraestructura de la nube, debido a que los requerimientos entre algunos módulos que componen la aplicación pueden ser contradictorios.
- Dificultan la escalabilidad, al escalar es necesario escalar toda la aplicación, lo que deviene en el consumo de grandes cantidades de recursos que normalmente no son utilizados

4.2 Arquitectura de microservicios

Un microservicio es una pequeña aplicación que se puede implementar de forma independiente, escalar de forma independiente y probar de forma independiente y que tiene una única responsabilidad [20]. Con responsabilidad única por un lado se refiere a que tiene una sola razón para cambiar y / o una sola razón para ser reemplazada. Pero el otro lado, una responsabilidad única en el sentido de que solo hace una cosa y una sola cosa y así puede entenderse fácilmente.

Una arquitectura de microservicios es una aplicación distribuida donde todos sus módulos son microservicios. Un análisis de las características más sobresalientes de una arquitectura de microservicios se puede encontrar en [19], a continuación se enumeran algunas de ellas:

- Los microservicios se pueden desplegar por separado permitiendo gestionar independientemente sus respectivos ciclos de vida.
- Nuevas versiones de microservicios se pueden introducir gradualmente en el sistema, desplegándolas junto a la versión anterior y los servicios que dependen de la última se pueden modificar gradualmente para interactuar con la primera. Esto fomenta la integración continua y facilita enormemente el mantenimiento del software.
- Como consecuencia del anterior ítem, desplegar cambios en un módulo de una arquitectura de microservicios no requiere un reinicio completo del sistema. Solamente se reinicia el microservicio afectado.
- Los componentes pueden ser más especializados, ya que pueden escribirse en diferentes tecnologías, siempre y cuando estas tecnologías admitan la interacción con las otras tecnologías utilizadas en la misma arquitectura de microservicios.
- Escalar una arquitectura de microservicio no implica una duplicación de todos sus componentes, y los desarrolladores pueden convenientemente desplegar o eliminar instancias de microservicios en función de la carga de trabajo que tengan en un determinado momento temporal.
- El camino para desplegar microservicios son los contenedores [21]. Esto le otorga al desarrollador un alto grado de libertad en cuanto a la selección y configuración del entorno de despliegue en la nube.

5 Marco de trabajo para desarrollar sistemas de programación de la producción

A continuación se presenta el marco de trabajo para desarrollar sistemas de programación de la producción utilizando servicios en la nube (ver Fig. 1). El marco está constituido por un conjunto de servicios integrados que tienen una alta cohesión y un bajo acoplamiento que facilita la construcción de un sistema de scheduling. La idea principal que guía el diseño del marco es el estilo arquitectónico de microservicios. Tal como se describió en apartado anterior este estilo permite concebir un sistema como una colección de servicios pequeños y autónomos. De esta forma cada servicio en el marco es independiente e implementa una funcionalidad bien definida de un sistema de

programación de la producción. Los microservicios se pueden comunicar de dos formas, con mensajes sincrónicos o mensajes asincrónicos. Los mensajes sincrónicos se utilizan cuando los microservicios esperan por una respuesta de otro microservicio para continuar con su tarea. Los mensajes asincrónicos se utilizan en situaciones donde los microservicios clientes no esperan por la respuesta de otro microservicio. El Gestor de Eventos es el encargado de la comunicación.

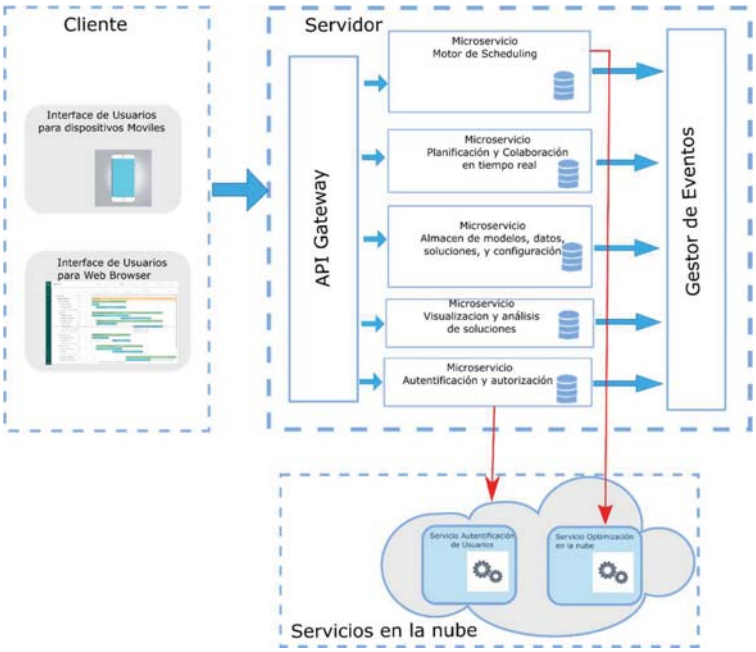

Fig. 1. Arquitectura del marco de trabajo para desarrollar sistemas de programación de la producción.

Los clientes, aplicaciones móviles o para browser, no llaman directamente a los servicios sino que lo hacen a través de una puerta de entrada que es interfaz de programación de aplicaciones, denominada "API Gateway" [22]. Esta puerta de entrada redirige la solicitud del cliente al servicio que atiende esta solicitud, desacoplando de esta manera la solicitud del cliente de los servicios que la atienden, de esta manera varios servicios pueden coordinarse para atender una determinada solicitud. Otra ventaja es que los servicios pueden ser modificados, versionados o refactoreados sin necesidad de modificar nada en el cliente. También permite que los servicios internamente utilicen otros protocolos de comunicación que no sean amigables con la web. Hay una serie de tareas (cross-cutting concerns) que pueden ser llevadas a cabo el API Gateway tales como, registro y descubrimiento de servicios, balanceo de carga, monitoreo, y seguridad, tolerancias a fallo. El *Microservicio Motor de scheduling*, es el microservicio núcleo del sistema. Este se encarga de obtener una solución a un

determinado problema de scheduling. En este contexto un problema de scheduling, en su caso más simple, queda definido por un archivo modelo y un archivo de datos que aplica al modelo. En un dominio dado es posible que para resolver un determinado problema de scheduling se requiera más de un modelo. En el lado del cliente se construyen las diferentes interfaces para el sistema de scheduling. Así, el proceso de planificación comienza con el usuario haciendo la carga de datos de un determinado problema, este proceso tiene lugar en el lado del cliente, ayudado por la interfaz de usuario, el resultado de esta tarea es remitido al servidor. El modulo API Gateway recibe la solicitud para resolver el problema enviado, analiza la solicitud y solicita al servicio *Microservicio Autentificación* las credenciales correspondientes para poder acceder a las API de los microservicios implicados en el proceso de obtener la solución al problema. Con las credenciales el API Gateway determina el tipo de solicitud del cliente y remite los datos al *Microservicio Motor de scheduling*, con los datos remitidos desde el API Gateway determina que archivos debe solicitar al *Microservicio Almacén de modelos, datos, soluciones y configuraciones*. Es posible que en las bases de datos del *Microservicio Almacén* ya exista una solución para el problema en cuestión, en este caso es comunicada y enviada al *Microservicio Motor de Scheduling* el cual envía el archivo de solución al *Microservicio de Visualización y análisis de soluciones* el cual confecciona los datos necesarios para los reportes y la visualización gráfica (diagrama de Gantt) que serán enviados al cliente. Pero si no existe tal solución, el *Microservicio Motor de Scheduling* remite los archivos modelo, datos, configuración, y credenciales al Servicio de Optimización en la Nube, el cual es un servicio externo y normalmente tarifado. Este servicio retorna un archivo con la solución al problema de scheduling, luego el *Microservicio Motor de Scheduling* envía este archivo de solución al *Microservicio de Visualización y análisis de soluciones*, y finalmente este retorna la solución en forma de reportes y gráficos al cliente.

El *Microservicio Autentificación y autorización* es el encargado de gestionar la seguridad del sistema, este autentifica los usuarios del sistema y determina a que parte del sistema están autorizados a acceder.

El *Microservicio planificación y colaboración en tiempo real*, en una arquitectura jerárquica ANSI/ISA 95 tiene por objeto asistir a la planificación a corto plazo, donde el horizonte de los scheduling no va más allá de los 10 días. Son scheduling que deben considerar más restricciones que los de planificación a largo plazo. Normalmente se trata de scheduling multiobjetivo en donde el cumplimiento de algunos objetivos se hace a costa de no cumplir otros y viceversa. Esta situación conduce a conflictos de intereses entre los departamentos involucrados en scheduling. Es aquí donde la colaboración en tiempo real para scheduling interactivo puede asistir a la búsqueda de consensos para establecer en un programa de producción consensuado. El grupo de usuarios involucrados puede ver e interactuar en tiempo real de web proponiendo o aceptando los cambios en el programa de producción. En un enfoque más moderno, es decir planificación y programación en industria 4.0, el CPPS (Cyber-physical production systems) es el encargado de las funciones del MES. En este tipo de contexto la planificación es del tipo Smart Scheduling [15]. Los eventos disruptivos que suceden en tiempo real en el proceso de producción deben ser atendidos por el sistema de scheduling. Para esta situación el *Microservicio* de *planificación y colaboración en*

tiempo real es el encargado del monitoreo de las actividades de producción en tiempo real y de elevar las peticiones de nuevos programas (re-scheduling) al *Microservicio Motor de Scheduling* con el objeto de cumplir los objetivos del programa de producción original.

6 Prototipando un sistema de scheduling mediante el marco propuesto

Con el objeto de ilustrar el uso del marco propuesto se describe a modo general, y sin entrar en detalle, la construcción de un muy simple sistema de scheduling utilizando el marco. La entrada al sistema de scheduling está compuesta por un conjunto de actividades, sus duraciones y sus precedencias. También se establecen los recursos que son necesarios para ejecutar las actividades, tales como máquinas y operarios. Las relación entre actividades y recursos se describe mediante un conjunto de restricciones tales como, una actividad *x* requiere del recurso *y* para ejecutarse. Por último, el objetivo del scheduling puede ser de diversa naturaleza, por ejemplo: optimizar el tiempo de entrega, u optimizar el uso de los recursos. Para desarrollar el sistema siguiendo el marco se proponen las siguientes tecnologías: Meteor.js [23] para implementar todos los microservicios. Mientras que los servicios en la nube de optimización y de autentificación son IBM Decision Optimization on Cloud [24] y Auth0 [25] respectivamente. MongoDB [26] como motor de base de datos para todos los Microservicios, y como Gestor de Eventos el mecanismo reactivo publicar / suscribir de Meteor.js. Ambassador como API Gateway [27]. Cada microservicio será desplegado en un contenedor Docker [28] y la gestión de contenedores se realizara con Kubernetes [29]. La plataforma cloud IaaS para despliegue de todo el sistema será digitalOcean [30]. El corazón del sistema es el *Microservicio Motor de scheduling* que es el encargado de transformar los datos que envía la interfaz de usuario (objetivo, actividades, recursos y restricciones) en la entrada que necesita el motor de optimización de IBM, luego, la respuesta enviada por el este motor, sirve para gestionar el trabajo de los demás microservicios. Meteor.js adhiere a la programación web reactiva por lo que resulta de gran ayuda en el desarrollo del *Microservicio de Planificación y Colaboración en tiempo real.* La gestión de la seguridad se realiza mediante tokens de acceso, los cuales brindan acceso a los diferentes microservicios, y son gestionados por la plataforma Auth0. Escalabilidad, balanceo, registro y descubrimiento de servicios y tolerancia a fallos son gestionados por Ambassador API Gateway.

7 Conclusiones

En este artículo se ha presentado un marco de trabajo para desarrollar sistemas de programación de la producción utilizando servicios en la nube. El marco hace uso de un conjunto de servicios de la nube entre en los que se destaca el motor de soluciones IBM Decision Optimization. Está basado en el estilo arquitectónico de microservicios y cuyas características han sido revisadas en el apartado 4.2. En el apartado 6 se ha

esbozado de manera muy general la aplicación del marco propuesto a la construcción de un sistema de scheduling en el contexto de una problemática común del dominio industrial. El objetivo de dicho apartado no ha sido demostrar como el marco puede ayudar a resolver una problemática en concreto, sino demostrar qué tecnologías pueden ser utilizadas en los diferentes módulos que este marco presenta. Una aplicación basada en el marco presentado está pensada para ser desplegada en cualquier plataforma de la nube que presente el modelo de servicios IaaS. Este tipo de plataformas permite aprovechar toda la potencia que brinda la nube en cuanto a escalabilidad, tolerancia a fallos, velocidad de cómputo, y economía de uso. Por último se debe mencionar que la arquitectura del marco ha sido diseñada en función de satisfacer algunas de las necesidades que presenta la industria 4.0, en lo que respecta a programación de la producción.

Como trabajos futuros se propone aplicar el marco de trabajo al desarrollo de sistemas de scheduling en una problemática industria real y concreta. En donde se puedan diseñar e implementar todos los módulos de la arquitectura y realizar diversos experimentos que pongan a prueba las bondades que brinda la nube. Sin duda el conocimiento obtenido de tales experiencias ayudará a mejorar el marco presentado.

Referencias

1. Barker, K.R., *Elements of sequencing and scheduling*. 1974, New York: John Wiley and Sons.
2. Yen, B.P.-C. and M. Pinedo. *On the design and development of scheduling systems*. in *Fourth International Conference on Computer Integrated Manufacturing and Automation Technology*. 1994.
3. Guizzi, G., S. Vespoli, and S. Santini. *On The Architecture Scheduling Problem Of Industry 4.0*. in *CIISE*. 2017.
4. Rossit, D. and F. Tohmé, *Scheduling research contributions to Smart manufacturing.* Manufacturing Letters, 2018. **15**: p. 111-114.
5. MacCarthy, B.L. and J. Liu, *Addressing the gap in scheduling research: A review of optimization and heuristic methods in production scheduling.* International Journal of Production Research, 1993. **31**: p. 59-79.
6. Henning, G.P., *Towards a software engineering approach to the deployment and implementation of scheduling systems*, in *Computer Aided Chemical Engineering*. 2017, Elsevier. p. 1405-1410.
7. Fowler, M. and J. Lewis, *Microservices a definition of this new architectural term.* URL: http://martinfowler. com/articles/microservices. html, 2014: p. 22.
8. Smith, F.S., *OPIS: A Methodology and architecture for reactive scheduling*. 1994: Morgan Kaufmann.
9. Le Pape, C., *Implementation of Resource Constraints in ILOG SCHEDULE: A Library for the Development of Constraint-Based Scheduling Systems.* Intelligent Systems Engineering 3, 1994. **3**(2): p. 55-66.

10. Entrup, M.L., *Advanced Planning and Scheduling Systems*. 2005: Advanced Planning in Fresh Food Industries: Integrating Shelf Life into Production Planning. Physica-Verlag HD.
11. Baheti, R. and H. Gill, *Cyber-physical systems.* The impact of control technology, 2011. **12**(1): p. 161-166.
12. Monostori, L., *Cyber-physical production systems: Roots, expectations and R&D challenges.* Procedia Cirp, 2014. **17**: p. 9-13.
13. Hermann, M., T. Pentek, and B. Otto. *Design principles for industrie 4.0 scenarios*. in *2016 49th Hawaii international conference on system sciences (HICSS)*. 2016. IEEE.
14. Leusin, M., et al., *Solving the Job-Shop Scheduling Problem in the Industry 4.0 Era.* Technologies, 2018. **6**(4): p. 107.
15. Rossit, D.A., F. Tohmé, and M. Frutos, *Industry 4.0: Smart Scheduling.* International Journal of Production Research, 2018: p. 1-12.
16. Antonopoulos, N. and L. Gillam, *Cloud computing*. 2010: Springer.
17. Taylor, G., *Definición de Cloud Computing por el NIST.* blog Microsoft Technet, 2010. **accedido 05/08/2019**: p. https://blogs.technet.microsoft.com/guillermotaylor/2010/08/25/definicin-de-cloud-computing-por-el-nist/.
18. Castiglione, F.C., *Virtualización en Entornos Cloud.* Tesis de grado 2017. **FCEFyN - UNSJ**.
19. Dragoni, N., et al., *Microservices: yesterday, today, and tomorrow*, in *Present and ulterior software engineering*. 2017, Springer. p. 195-216.
20. Thönes, J., *Microservices.* IEEE software, 2015. **32**(1): p. 116-116.
21. Kovács, Á. *Comparison of different Linux containers*. in *2017 40th International Conference on Telecommunications and Signal Processing (TSP)*. 2017. IEEE.
22. Montesi, F. and J. Weber, *Circuit breakers, discovery, and API gateways in microservices.* arXiv preprint arXiv:1609.05830, 2016.
23. Meteor, *Meteor documentation.* www.meteor.com, 2019.
24. DOcplexcloud, *Decision Optimization on Cloud.* https://developer.ibm.com/docloud/documentation/decision-optimization-on-cloud/, 2019.
25. Auth0, *Auth0 Platform* https://auth0.com/, 2019.
26. MongoDb, *Motor de base de datos basados en documentos.* https://www.mongodb.com/, 2019.
27. Ambassador, *API Gateway* https://www.getambassador.io/ 2019.
28. Docker, *Enterprise Container Platform.* https://www.docker.com/, 2019.
29. Kubernetes, *Gestor a escala de aplicaciones en contenedores.* https://kubernetes.io/es/, 2019.
30. DigitalOcean, *Proveedor de servidores virtuales para nube mediante el modelo de servicio IaaS.* https://www.digitalocean.com/, 2019.