J Intell Manuf (2016) 27:447–462
DOI 10.1007/s10845-014-0878-7

Shop-floor resource virtualization layer with private cloud support

Octavian Morariu · Cristina Morariu · Theodor Borangiu

Received: 19 August 2013 / Accepted: 17 January 2014 / Published online: 30 January 2014

Abstract Large scale emergence of mature cloud solutions, ranging from software-as-a-service based solutions for business management, to very sophisticate private cloud solutions; offer the building blocks for constructing extremely flexible enterprise systems that can respond to environmental changes with great agility. Manufacturing enterprises need to adopt these new technologies to advance in a new era of mass customization where flexibility, scalability and agility are the differentiating factors. In this context, this paper introduces the virtualized MES and shop floor architecture as an intermediate layer in the manufacturing stack and discusses the advantages offered by this approach for manufacturing enterprises. A classification of MES and shop floor devices is presented focusing on the virtualization techniques suitable for each device type, considering the level of distributed intelligence and the virtualization overhead. Shop floor virtualization through shop floor profiles is presented and discussed underlying the flexibility of the solution. A pilot multi-agent implementation for virtual shop floor configuration based on the CoBASA reference architecture is presented and discussed. The shop floor profiles which define the virtual layout and mappings of the robotized manufacturing system are also provided in this context. The pilot implementation using six Adapt robots and a IBM CloudBurst 2.1 private cloud, is described and virtualization overhead in terms of event propagation delays is measured and presented in several scenarios of resource workload collocation on physical cloud blades

Keywords MES · Robot virtualization · Private cloud · Shop floor · Virtual devices · Mapping · Shop floor profile · High performance computing

O. Morariu (✉) · T. Borangiu
Centre for Research and Training in Industrial Control, Robotics and Materials Engineering, University "Politehnica" Bucharest, Bucharest, Romania
e-mail: octavian.morariu@cimr.pub.ro

T. Borangiu
e-mail: theodor.borangiu@cimr.pub.ro

C. Morariu
Cloud Computing Research Department, CloudTroopers International, Cluj-Napoca, Romania
e-mail: cristina@cloudtroopers.ro

Introduction

The large scale emergence of cloud computing platforms bring unprecedented opportunities for manufacturing enterprises to reduce operational costs and at the same time to embrace cutting edge technology advances, especially in the information management realm. Like with most business areas, cloud computing has the potential to deliver its promised value if the adoption is done diligently and the migration from legacy systems to cloud platforms is planned carefully (Leimeister et al. 2010). For manufacturing systems, the cloud adoption potential translates in five main factors: cost reduction, increased flexibility of the system, fault tolerance, scalability and agility. These factors derive from the virtualization of workloads which offers a new layer of instrumentation both during implementation phases of the system and at runtime.

A section view through a typical manufacturing enterprise would show a layered architecture for information flow and management. The top layer is usually consisting in a set of business processes or service choreographies living in a service oriented architecture (SOA) ecosystem. SOA represents the materialization of the component based architecture, in terms of separation of concerns, functionality and interaction definitions. At SOA level the individual business functions or services, are aggregated and composed using concepts as

service choreographies and orchestrations. A choreography is a form of de-centralized collaboration between services, representing business components. Successful manufacturing enterprises understood that SOA adoption and alignment to industry standards is a mandatory step in reducing operational costs and represents the only alternative to remain competitive in a globalized economy (Morariu et al. 2013). SOA governance is by now achieved and established inside leading manufacturing enterprises and the main focus now shifts to key performance indicators (KPI's) analysis, process refinement and continuous improvement. The top layer contains also the main integration points with third party suppliers of materials and services. The middle layer is represented by the manufacturing execution system (MES) also known in the literature as MES layer, which is responsible with information breakdown and aggregation of actual operations required for manufacturing of the products. Traditionally MES systems are proprietary implementations, either as monolithic control architectures or multi-agent systems, and are responsible for the complete control of the shop floor activities: product and operation scheduling (Babiceanu and Chen 2006), events processing (Thomas et al. 2012), production tracking (Zhang et al. 2012b) and so on (Leitão et al. 2012; Witsch and Vogel-Heuser 2012). Finally the shop floor represents the physical layer where robots, conveyors and intelligent products are interacting and specific operations are performed. The first two layers mainly deal with information flow, while the shop floor layer deals with material flow (Mason-Jones and Towill 1997).

Recent research outlines the opportunities available for manufacturing enterprises when it comes to adoption of cloud computing concepts (Lohr 2007; Vecchiola et al. 2009; Pallis 2010). The main trend is to identify services at the top layer that can be partially or completely externalized or replaced with applications offered as a service by cloud providers. A classical example (Weissman and Bobrowski 2009) is Salesforce.com, on the customer facing activities and on financial applications, that was able to replace entire departments and legacy applications in many manufacturing enterprises with a fast return of investments (ROI) and significant operational process improvement and customer satisfaction. Other examples are also available, where migrating from legacy applications and processes to software-as-a-service (SaaS) proved to be a good strategic decision for enterprises. However, not every aspect of the information flow can be migrated towards a public cloud platform, for various reasons. The most relevant are related to the dependency on the location where the information resides and is processed (Svantesson and Clarke 2010), security concerns (Kaufman 2009; Jamil and Zaki 2011), legal constraints (Pearson and Benameur 2010) and so on. In this context, private cloud solutions can add significant benefits for manufacturing companies, especially when considering the MES and shop floor layers, by providing workload virtualization.

This paper introduces the virtualized MES and shop floor architecture as an intermediate layer in the manufacturing stack and explores the advantages offered by this approach. "The business case of virtualization and private clouds" section describes the business case for private clouds and the general characteristics of this class of systems. "Related work" section provides an overview of the related work on the usages of cloud computing concepts in manufacturing enterprises and outlines the relation of our contribution in this context. In "MES and Shop Floor Virtualization" section the MES and shop floor virtualization concepts are discussed in detail and in "Application example: multi-agent MES based on CoBASA architecture" section, an application of the virtualization approach is exemplified on CoBASA reference architecture. "Pilot implementation and experimental results" section presents the pilot implementation using a six station manufacturing cell and an IBM CloudBurst 2.1 private cloud. The virtualization benefits and the final conclusions are presented in the final section together with future research directions.

The business case of virtualization and private clouds

Cloud computing is a term that is used to describe any system that is delivering hosted computing services over the Internet. Cloud computing represents a new model of delivering IT services to end users and enterprises by providing computation power, software, data access and storage on demand. The main concept is that cloud providers are offering computation as a utility, similar to the concept used by electricity grids to provide energy to end users. The concept of cloud computing was first mentioned back in 1960s by John McCarthy but because of technological limitations of the computers at the time, his vision never got enough traction in the industry to become a reality. Since then, there were several technological breakthroughs that allowed the emergence of cloud computing.

National Institute of Standards and Technology (Mell and Timothy 2011) has defined the main cloud characteristics, as follows:

- *On-demand self-service* A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider.
- *Broad network access* Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).

- *Resource pooling* The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.
- *Rapid elasticity* Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.
- *Measured service* Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported providing transparency for both the provider and consumer of the utilized service.

Additionally we can consider other cloud characteristics:

- *Multi-tenancy* refers to the fact that the same resource instance is being used concurrently by multiple client organizations. This can be seen at all three layers discussed, because in each model there is some resource being shared, either the actual hardware box, the hypervisor, the platform or the actual software application. This is a huge advantage in terms of resource usage optimization, but raises a number of challenges, specially related to customer data security, which will be discussed in detail later in this book.
- *Pay-per-use* model refers to the cost model adopted by the cloud providers. This allows a low start-up investment from customers at the same time allowing them to use high end technologies.

One of the most important adoption drivers for cloud computing is the conversion of traditional computing systems to virtualized environments. Virtualizing a computing system means that hardware and software resources are organized and managed in a pool model, allowing multiple beneficiaries to use these resources as needed. This approach leads directly to better resource utilizations, which represents the primary goal and benefit for any form of cloud computing.

Historically computer systems, like databases, BI servers, and so on, were deployed on dedicated physical infrastructures, or in other words, dedicated servers, storage and usually dedicated network as well. Some of the most common reasons for this approach were the poor standard adherence on legacy implementation, strict certification margins imposed by software vendors, no real option to impose hard resource utilization quotas at operating system level, and so on. The result of this practice was that the overall resource utilization was below 20 % on average (Lee and Zomaya 2012). The pressure to reduce costs and optimize resource utilization, in this context, has set the scene for a fundamental change towards virtualization as a strategy, from CIOs of large enterprises.

The advances in virtualization technologies allows creating virtual instances of various physical devices present in an computing environment, like CPUs, memory, storage, networking, USB ports and various other devices as required. Virtualization is achieved by introducing an intermediate software component between the physical resources and the operating system layer called hypervisor (Gavrilovska et al. 2007; Mergen et al. 2006; Raj and Schwan 2007). A hypervisor is transparent for the guest operating system and offers virtual resources that are mapped to the physical layer. Over the last years, hypervisors have evolved and provide reach features allowing fine grained management of physical resources together with many real time optimizations.

Private clouds offerings are targeted to the scenarios where the enterprise strategic direction is to keep the information storage and processing inside the premises, but still to take advantage of virtualization and improve resource usages. Some of the direct advantages of moving from legacy computing environments to a private cloud architecture is the reduction of power consumption, data centre floor space and cooling requirements. From a technical point of view, private clouds can be constructed from existing computing resources by using a cloud management stack, or can be purchased as turn-key solutions from cloud vendors like IBM, Oracle, Microsoft, RedHat and others. The commercial solutions typically come as a set of blade servers, dedicated storage devices, network devices and a cloud management software stack that implements the workload discovery and service catalogue, provisioning workflows and basic administration of resources.

For manufacturing enterprises these various cloud offerings ranging from SaaS based solutions for business management, to very sophisticate private cloud solutions; offer the building blocks for constructing extremely flexible enterprise systems that can respond to environmental changes with great agility (Wang et al. 2012). Manufacturing enterprises need to adopt these new technologies to advance in a new era of mass customization where flexibility, scalability and agility are the differentiating factors for real-time control manufacturing systems.

Related work

Cloud manufacturing (CMfg) was introduced (Li et al. 2010) as a service-oriented networked manufacturing model, focusing on studying the opportunities for networked manufacturing (NM) opened by the cloud computing platforms. The cloud based service delivery model for the manufacturing realm, like product design, batch planning, product scheduling, real time manufacturing control, testing, management, and all other stages of a product life cycle were described by Xu (2012). Similar concepts were discussed by Zhang et al. (2012a) focusing on typical characteristics of CMfg and the key technologies for implementing a CMfg service platform.

Research by Wang and Xu (2013) introduced Interoperable Cloud-based Manufacturing System (ICMS). ICMS provides a Cloud-based environment integrating existing and future manufacturing resources by packaging them using the Virtual Function Block mechanism and standardized descriptions. The resource sharing using cloud computing was also explored by Wu and Yang (2010) focusing on providing cooperative work between enterprises for global manufacturing. Another research group (Cheng et al. 2010) provides a comprehensive study on the utility model and utility equilibrium of resource service transaction in CMfg.

Another important research area is represented resource virtualization techniques and resource sharing in manufacturing environments (Wu and Yang 2010). Shi et al. (2007) proposed a model for resource sharing in grid manufacturing. The framework proposed is composed of a network infrastructure layer, a manufacturing resource aggregation layer, a manufacturing resource management layer, a manufacturing service application layer and a portal layer. A manufacturing resource hierarchy model (MRHM), which consists of a manufacturing resource layer, a resource expressing layer and a resource interface layer, is proposed and discussed. In the same direction, Liu and Li (2012) argues that resources and resource capabilities virtualization and modeling are the starting point for manufacturing cloud services encapsulation. The team provides a manufacturing resource virtual description model that includes both nonfunctional and functional features of manufacturing resources. Additional research done by Luo et al. (2013) expands the concept and provides a modeling and description method of multidimensional information for manufacturing capability in CMfg system.

The CMfg and resource virtualization concepts presented in the above cited articles are focusing on inter enterprise cooperation for optimized manufacturing using the emerging cloud technologies as enabler technology. While we agree that the CMfg concepts are correct and represent the future of manufacturing, we argue that MES and shop floor virtualization concepts described in this paper are prerequisites for effective collaboration through cloud technologies. The concepts introduced in this paper are meant to fill the gap between the theoretical concepts behind CMfg and the current shop floor challenges in the context of virtualization adoption inside manufacturing enterprises.

MES and shop floor virtualization

A cross section through the IT landscape of a manufacturing enterprise would show typically a three layered architecture as illustrated in Fig. 1. The architecture must be robust enough to assure the information flow is in sync with

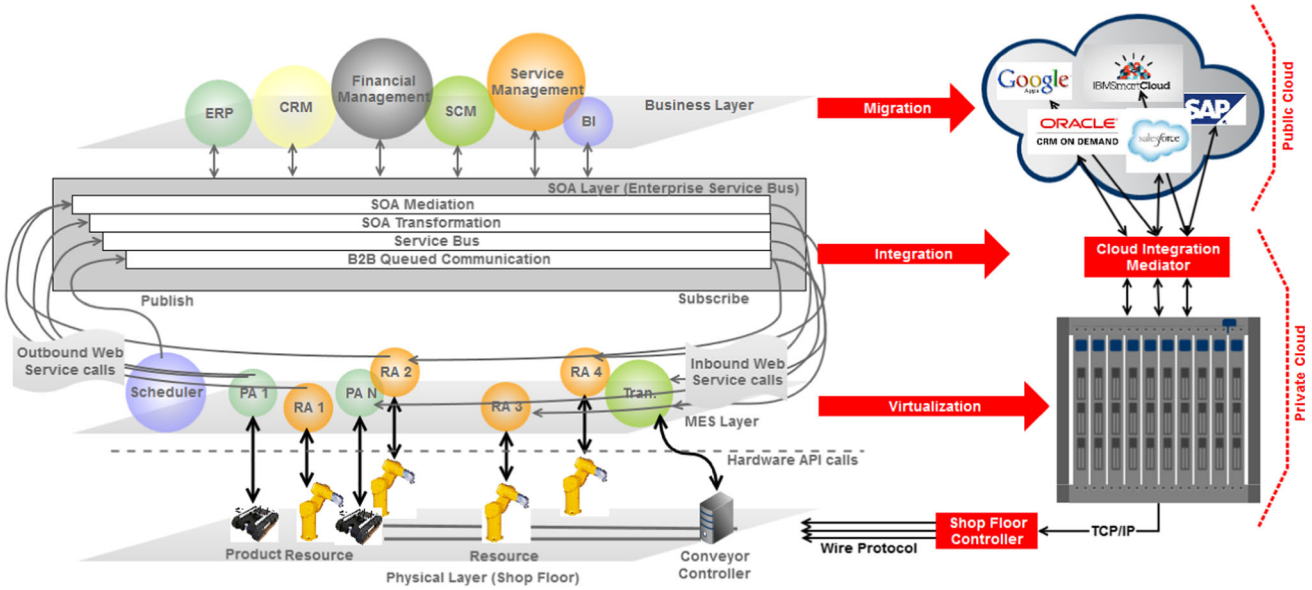


Fig. 1 From service oriented manufacturing systems to cloud

the material flow at all times, and flexible enough to allow dynamic reconfiguration and SOA governance. One practical example in this direction is the IBM Manufacturing Integration Framework (Morariu et al. 2012), implemented successfully in automotive and electronics industries by several large organizations. Similar SOA based solutions targeted for manufacturing enterprises are available from other software vendors and integrators.

The lower layer consists in the so called shop floor or physical layer and contains the resources, transportation devices and products involved in the manufacturing process. The physical software workloads at shop floor layer are proprietary (custom hardware and custom software) or semi-proprietary (standard hardware and operating system and custom software) and are responsible with direct control of the physical resources and of the material flow on the shop floor. Tightly coupled with the physical layer is the MES. The MES consists in a set of workloads directly connected to the physical resources that can be seen as agents representing the resource or product (in Fig. 1 illustrated as Resource Agent x/RAx, Product Agent x/PAx). Also, the MES layer contains a shop floor scheduler and a transportation manager for the conveyor.

The communication pattern between MES / physical workloads is, from a SOA perspective, a service choreography with point to point message exchange in real time. When considering migration to cloud environments, these characteristics of MES/physical layer can be assured by private cloud implementation with *workload virtualization*.

At the heart of the service oriented manufacturing systems is the enterprise service bus/manufacturing service bus (MSB) coupling. The Enterprise Service Bus was first proposed by Chappell (2009) a software architecture that has a set of key characteristics:

- Message routing and control across enterprise components
- Decoupling of various modules by asynchronous messaging, replacing point to point communication with the common bus architecture
- Promote reusability of utility services, reducing the number of redundant services across the enterprise
- Provide transformation and translation of messages to allow easy integration of legacy applications
- Provide an engine for workflow execution

As of this date, all major software companies provide solid commercial implementation of ESB on top of their SOA offerings: IBM WebSphere ESB, Microsoft BizTalk Server, Oracle Enterprise Service Bus. Along with commercial implementation there are also open source solutions: JBoss ESB, Open ESB, Apache ServiceMix and others.

The MSB integration model is an adaptation of ESB for manufacturing enterprises. We have identified the following main characteristics of a MSB, in addition to the ones inherited from the ESB:

Event driven communication: At shop floor level, during manufacturing process there are a high number of events generated that need to be handled by specific components. For example, when a pallet arrives in a given position on the conveyor belt, a sensor detects the associated RFID tag and generates an event. This event needs to be dispatched to the relevant resources in order to be processed by the scheduling module or by the actual robot that performs an operation. The main role of the MSB implementation is to perform the event dispatch operation allowing shop floor components to exchange information in an event driven fashion.

Workflows: Along with event dispatching, the MSB has the ability to launch and execute predefined workflows associated with specific events. Workflows consist in a set of successive operations, either automated or manual (human interventions). Workflows are typically required for exceptional events that require complex logic to handle them, like un-expected resource break-downs or rush orders. Another advantage of grouping complex logic in workflows is that the workflows are "external" in reference to the MSB, from the implementation point of view.

Message transformation: The shop floor level integrates a wide range of modules, from software schedulers to various hardware devices (robots, sensors, etc.). From a communication perspective, the protocols and the message formats used can be a simple +5V DC signal, proprietary line protocols or event high level TCP based protocols. The MSB role is to transform these messages to and from these proprietary protocols in a common standardized format. This is done by allowing the development of message convertors at any entry point and exit point of the bus.

Synchronous and asynchronous communication: The MSB implementation offers both synchronous and asynchronous communication models. The synchronous model causes the sender of the message to block until the response is received and so is implicitly bidirectional. The asynchronous model is using a queue based mechanism, where the sender submits the message and from where the receiver picks it up at a later time. This allows decoupling of the execution of the sender and the receiver. At the shop floor level both communication models are useful.

Message persistence: When asynchronous model is used, the messages reside in logical queues from where they are consumed. The MSB implementation stores the queues in a persistent highly available storage that allows production state recovery in case of a system crash. The MSB can use a network file system or a distributed database as a repository for the message queues.

In a cloud based implementation, the MSB layer with the above mentioned characteristics becomes a distinct component called Cloud Integration Mediator, bridging the gap

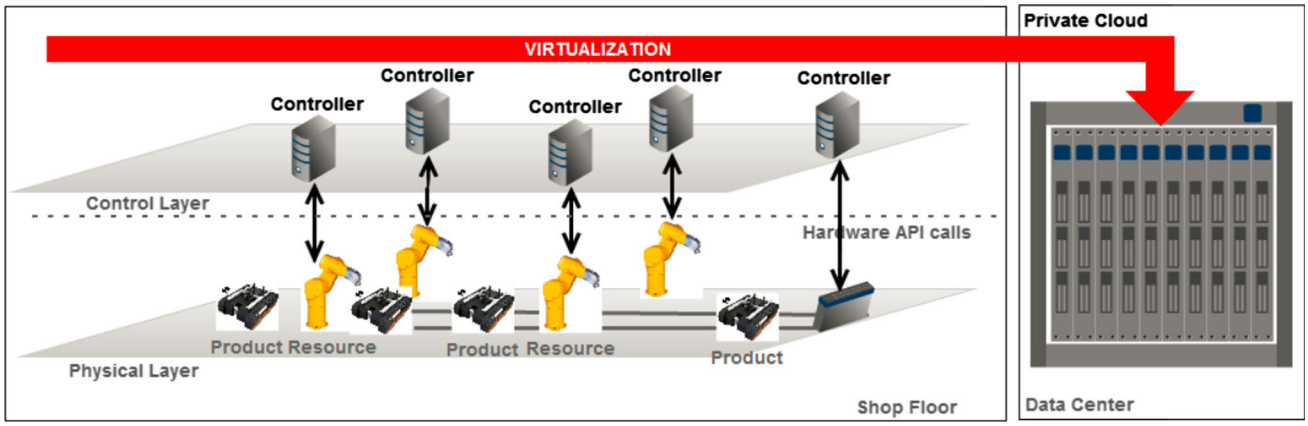


Fig. 2 From shop floor workloads to data centre

between the local private cloud implementation and the public cloud services used in a SaaS fashion. The mediator is conceptually built on top of SOA and must have ESB and B2B capabilities to integrate with internet provided applications. Implementing the mediator module is fundamentally an *integration effort*.

The information flow in a manufacturing enterprise is governed by business applications handling customer orders, supply chain operations, business intelligence and so on. These applications are traditionally operated and maintained by the company's own IT department using dedicated hardware stacks. This approach requires huge up front investments to setup the systems, high operational costs and most of times deliver bellow the intrinsic capabilities of the software products.

The large scale availability of applications delivered on a SaaS model raises an important opportunity for manufacturing enterprises to *migrate* from in-house implementations of complex business applications, to public cloud solutions available either in multi-tenant mode or dedicated hosted solutions.

In this context, the basic concept of MES and shop floor virtualization is illustrated in Fig. 2 and involves migration of all workloads that were traditionally executed on physical machines located on the shop floor to the data centre, specifically to the private cloud infrastructure as virtual workloads. The idea is to run all the control software in a virtualized environment and keep only the physical devices on the shop floor. This separation between hardware and software provides high flexibility and agility to the manufacturing solution.

Depending on the manufacturing system design and implementation, there can be several types of workloads that can be identified. The following section contains a classification of these workloads done from a virtualization perspective.

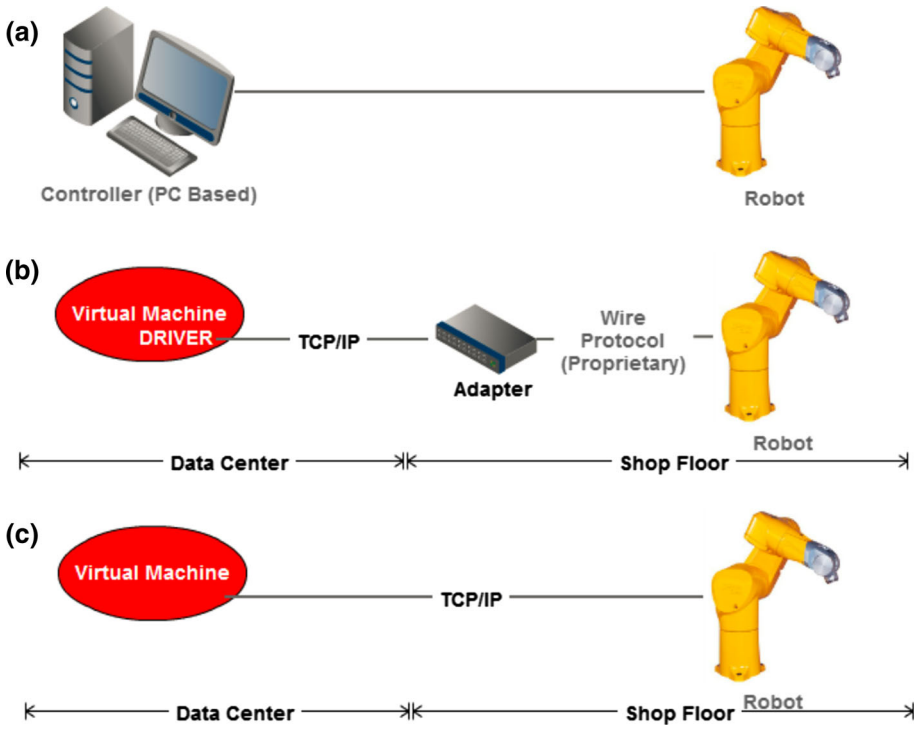
Virtualization of shop floor resources

Shop floor resources are active resources like robots, CNC machines and so on. The control architecture can vary depending on the manufacturer and technology used, but in general the resource is controlled by a PC based workstation. The communication between the control workstation and the physical resource can be either standard TCP/IP based, or a proprietary wire protocol. Figure 3 illustrates the approaches for the virtualization of these resources; (a) is the initial state without virtualization, while (b) and (c) are the two alternatives to workload virtualization. In case the resource can be accessed by TCP/IP directly, the virtualization is consists in virtualizing the workload directly and mapping a virtual network interface to it, which will be used to control the resource. However, in case a proprietary wire protocol is used, the virtualization process is more complex, as it would involve a local controller on the shop floor that would provide the physical interface for the wire protocol.

This physical interface would be virtualized and mapped through a specific driver to the virtualized workload over the network. From an implementation point of view, when standard communication protocols are used by the shop floor resources, the virtualization process is strait forward and would not require additional development. On the other hand, if proprietary protocols are used, the virtual workloads would require custom developed drivers to communicate with the shop floor controllers mapped over the network.

This scenario introduces the need for a new type of device on the shop floor, which would act as an extension of the private cloud system. This device hosts the physical communication interfaces for the shop floor resources and provides virtualized mapping for the workloads in the cloud. Figure 4 illustrates this concept.

The actual representation of the virtualized shop floor resources is done in XML format and referred to from the



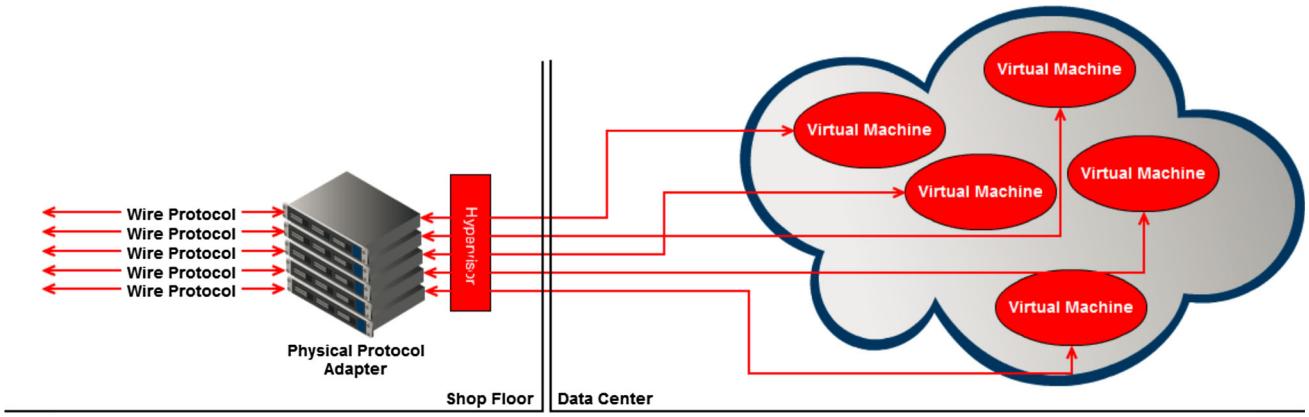


Fig. 3 Shop floor resource virtualization

Fig. 4 Physical protocol adapter architecture

shop floor profiles. This is described in the next sections of this chapter.

Virtualization of intelligent products

The requirements for manufacturing agility on one hand and supply chain predictability on the other hand converged and the concept of intelligent product (IP) has emerged. McFarlane et al. (2002) present the main characteristics of the intelligent product as the ability to monitor, assess and reason about its current and future state. At the same time, recent research offers several advances in supporting applications architectures especially towards SOA orientation, (Borangiu 2009) and MSB, (Morariu and Borangiu 2012).

Meyer et al. (2009) presents a complex survey on the intelligent product focusing on the underlying technologies that enable this concept. A classification is introduced that positions an intelligent product based on three directions: (1) level of intelligence, (2) location of intelligence, and (3) aggregation level of intelligence. The level of intelligence can be basic information handling, problem notification and even decision making. The location of intelligence can be either local, on the product itself, or remotely accessed through a network. Finally the aggregation level of intelligence refers to the granularity at which intelligence can be considered: individual product, product batch, container and so on.

The physical part of an intelligent product is composed from a product pallet and the product itself. The product pal-

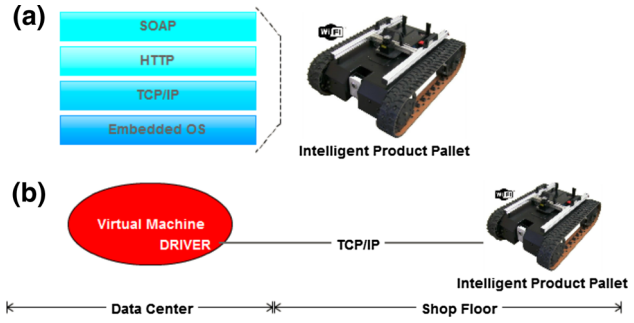


Fig. 5 Intelligent product virtualization

lets are usually equipped with devices able to run Data and CPU intensive applications in order to implement an intelligent behavior. For example they are able to run a full Java Virtual Machine on top of the embedded OS and have enough memory and processing power to be able to execute complex algorithms allowing them to make intelligent decisions, such as genetic algorithms for scheduling, Neural Networks for decision making and so on. In this category we can include mostly the Android equipped devices which can execute complex Java based agents. Among most utilized multi-agent platform is Java Development Environment (JADE) (Bellifemine et al. 2001). This class of devices represent the building blocks for a genuine distributed intelligence architecture, in which complex negotiation logic can be implemented at lower levels in the stack, allowing local decisions in the manufacturing process. These devices can leverage higher layer standards on top of SOAP (SOAP Standard, available online, 2013, www.w3.org/TR/soap) like ebXML (ebXML Standard, available online, 2013, www.ebxml.org), STEP (STEP Standard, available online, 2013, www.steptools.com) or OAG BOD (OAG BOD Standard, Open Applications Group Integration Specification (OAGIS), available online, 2013, www.oagi.org).

Virtualization for intelligent products is recommended when there is a large amount of data processing implemented on the product itself. Virtualization would allow moving the processing from the product pallet device to the cloud environment, either in a dedicated workload or in a shared workload (Fig. 5). The shared workload model is best suited for multi-agent system based MES implementations. The virtualization process would imply the mapping of the physical sensors and actuators installed on the product pallet to the virtual machine by using a thin hypervisor on the product pallet and WIFI network connection.

Workload management

The virtualization process starts with workload discovery and publication in the service catalogue. In practice this is done in several steps:

Step1 Identification of main characteristics of the physical controller for each resource on the shop floor, like hardware requirements, operating system, software stack, manufacturing specific control software, certifications of each on virtualized environment. This includes mainly the robot controllers and conveyor controller.

Step2 Creation of the virtual machine templates according to the specifications determined in the previous step. These virtual machines will contain the complete software stack required to control the physical resources, including the required drivers for special device mappings (i.e. wire protocol controller).

Step3 Creation of virtual machine templates for every MES component, like product scheduler, MSB and other specific control components.

Step4 Creation of the shared virtual machine template or dedicated virtual machines templates for the intelligent products, depending on the technology used and the level of distributed intelligence present in the manufacturing system.

Step5 Publication of all these virtual machine templates in the service catalogue, so that it can be deployed in the private cloud infrastructure as actual workloads.

Step6 Populating the resource catalogue with the resource definitions. There are two types of resources in the resource catalogue: cloud resources, like CPU, memory installed on the virtualization blades, installed storage, network devices and so on, and shop floor resources, like virtual protocol adapters for scenarios where proprietary wire protocol is used, virtualized robots, sensors and actuators from the product pallet, and so on. This step is done manually by an engineer having the role of manufacturing *system configurator*. At this stage, all the building blocks for the virtualized MES and shop floor are present in the service catalogue of the private cloud infrastructure. This six step process is illustrated in Fig. 6.

As the illustration suggest, this is an ongoing process as virtual workloads have to mirror the lifecycle of the physical shop floor resources. An example of a typical virtualization process that was achieved in the pilot implementation is described in "Virtualization process" section of this paper.

Shop floor profiles

The binding between workload templates and virtualized resources is done using *shop floor profiles*. Shop floor profiles are in fact XML files and contain a complete or partial definition of the manufacturing system virtual layout and mappings.

The shop floor profile is workload centric and basically contains a list of workload definitions, or in XSD:

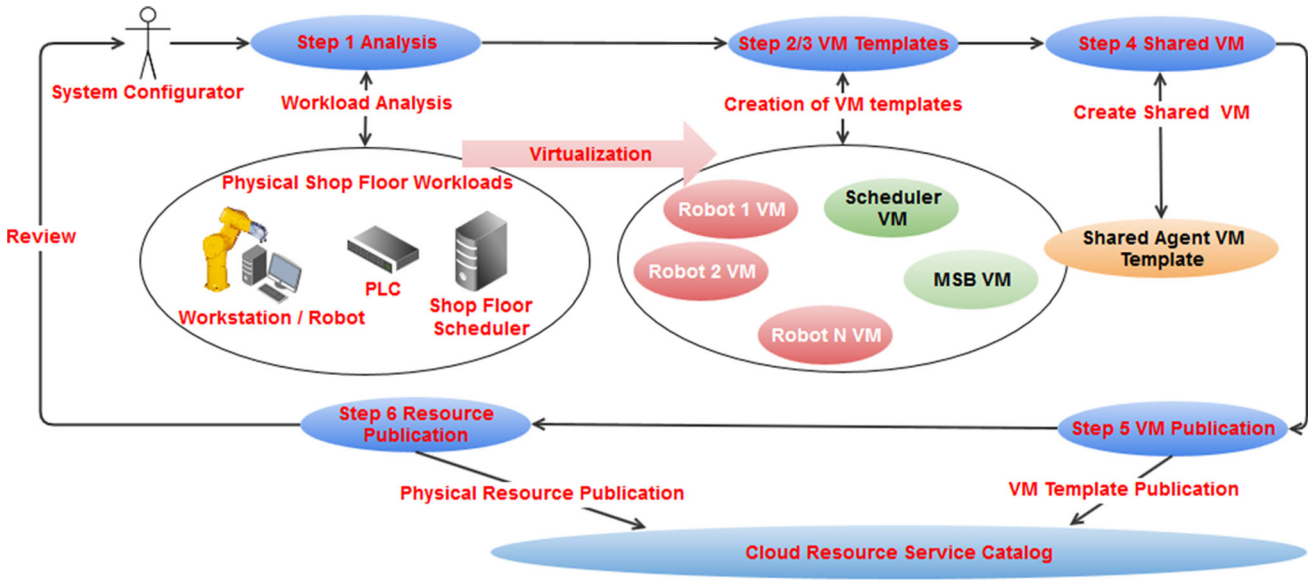


Fig. 6 Workload management process

```
<sequence>
<element name="ProfileName"
type="string"></element>
<element name="Revision"
type="int"></element>
<element name="Workloads" type="tns:Workload" minOccurs="1"></element>
</sequence>
</complexType>
```

The workload definition XSD specification is:

```
<complexType name="Workload">
<sequence>
<element name="Workload_ID" type="string">

</element>
<element name="VM_Reference" type="string">
</element>
<element name="VM_Revision" type="int">
</element>
<element name="Virtual_CPU" type="int">
</element>
<element name="Virtual_RAM" type="int">
</element>
<element name="Virtual_DISK" type="int">
</element>
<element name="Virtual_RESOURCES"
type="Resource">
</element>
</sequence>
</complexType>
```

The workload refers to a specific revision of a VM published in the service catalogue, a number of mapped virtual CPU cores, the amount of RAM memory allocated to the VM and the amount of Disk space. The workload also contains references to a list of mapped resources, together with parameters passed. The virtual resource XSD specification is:

```
<complexType name="Resource">
<sequence>
<element name="Resource_ID" type="string">
</element>
<element name="Resource_Type" type="string">
</element>
<element name="Resource_Description"
type="string">
</element>
<element name="GenericAttribute1" type="string">
</element>
<element name="GenericAttribute2" type="string">
</element>
<element name="GenericAttribute3" type="string">
</element>
<element name="GenericAttribute4" type="string">
</element>
<element name="GenericAttribute5" type="string">
</element>
</element>
</sequence>
</complexType>
```

The resource specification contains along with ID and type references, a set of generic attributes. These attributes have different meaning depending on the resource type. A relevant example for such an attribute is the IP address of a given network interface for a specific robot on the shop floor.

Shop floor profiles can also be nested by XML inclusion. This is useful when two or more workloads are defined as a pair. A practical example would be when two robots are placed together in the same workspace on the shop floor and can perform collaborative tasks and operations. In this scenario is easier to manage the two corresponding workloads

and resource mappings in a sub-profile that can be included in various shop floor profiles and so, reused many times.

Provisioning manager (PM)

The shop floor profiles are loaded by the *provisioning manager* (PM) component. The PM is responsible for parsing the shop floor profiles and creates the workload instances based on their definition, in the private cloud environment. The PM also maps and binds the virtualized resources to the VMs deployed in the cloud, running on the virtualization blades by using either standard network drivers, for TCP/IP accessible resources or by using custom drivers for proprietary communication protocols. To do so, the PM calls the hypervisor APIs. This concept is illustrated in Fig. 7.

The PM loads all the provisioning profiles available but only one main shop floor profile is active at a time, together with all the nested sub-profiles. Switching between shop floor profiles requires:

Step1 Saving the state of all workloads running in the current shop floor profile.

Step2 De-allocating the virtual resources from the workloads and returning them to the resource pool.

Step3 De-provision of the workloads from the virtualization blades and store them in suspended mode.

Step4 Loading the workloads defined in the new profile, either from the VM catalogue if the profile is executed for the first time, or from their suspended state, if the profile was executed before.

Step5 Allocating the virtual resources to each workload.

Step6 Starting the VMs or restoring their state on the virtualization blades.

The feature of the PM that allows quick switching between shop floor profiles provides the manufacturing system with a high level of flexibility, especially during the initial configuration phases where numerous tests are required, and during the configuration modification like, new resources added, shop floor changes, regular maintenance or unexpected resource break-down.

Another important feature that the PM can provide is the monitoring of the VM statuses and fail-over assurance. A workload crash can be detected and the associated VM can be restarted or re-provisioned, reducing the manufacturing system downtime at minimum.

Application example: multi-agent MES based on CoBASA architecture

The application example in this section explains how an existing MES implementation, in this case the CoBASA architecture, could be virtualized, showcasing how the concepts and workloads from the physical implementation map to the private cloud virtual implementation.

The CoBASA architecture (Barata and Camarinha-Matos 2003) introduces an agent-based control architecture in which cooperation regulated by contracts is proposed as a flexible approach to dynamic shop floor re-engineering. It describes the dynamic and flexible cooperation of manufacturing agents representing resources (here robots), and how they can be created from a generic agent template. The flexibility is assured by the resource (robot) consortium concept defined in the CoBASA architecture. In our implementation example below, the PM module creates a direct mapping between the shop floor profiles for virtualization and the Manufacturing Resource Agent (MRA) consortium (robot team), as shown in Fig. 8. The PM interacts with the MRAs to gather real time information on robot availability and status and with the Customer Order Management (COM) module to retrieve the customer order. Based on the product types specified in the customer order, the PM generates a list of operations required for the manufacturing of the product batch and selects the MRAs representing the robots capable of performing the operations. Once MRAs selected and the hierarchical

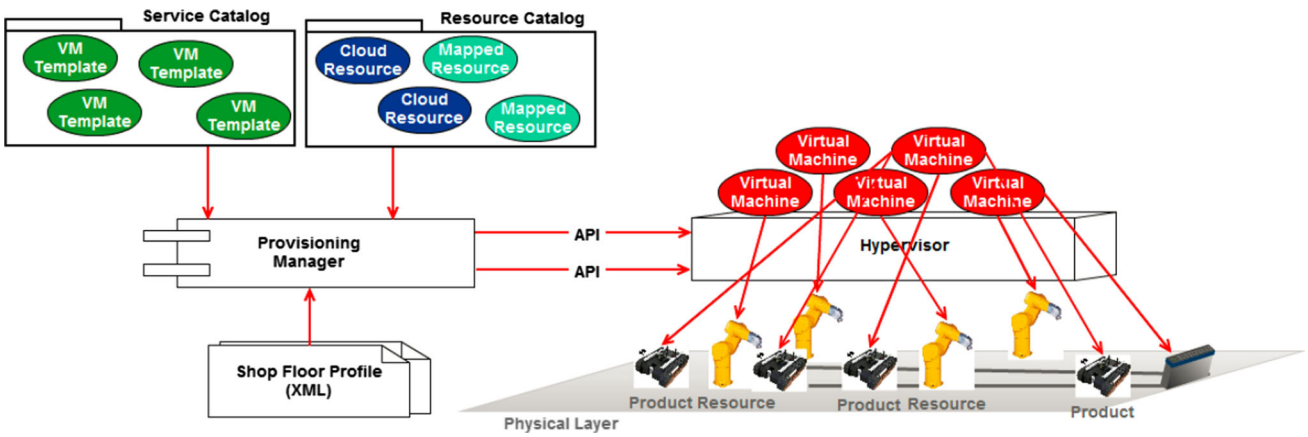


Fig. 7 Provisioning manager, provisioning profiles

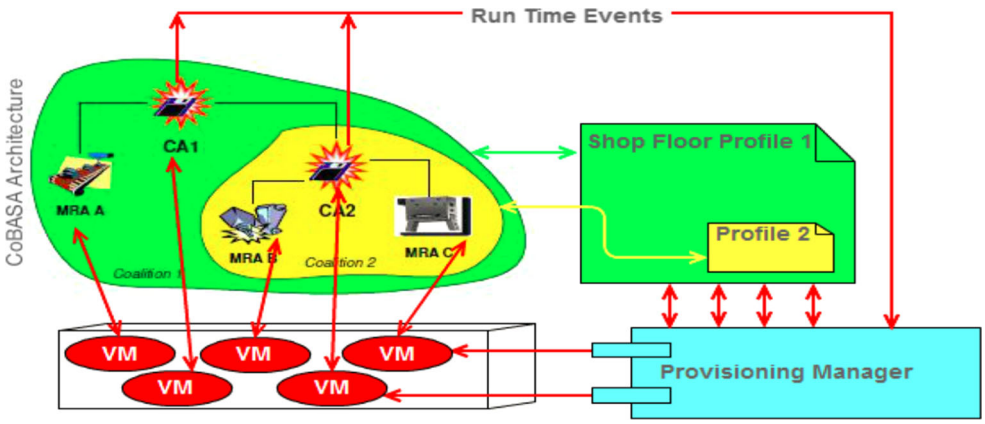
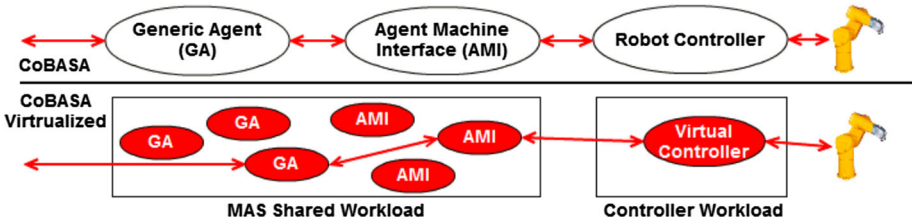


Fig. 8 CoBASA mapping with shop floor profiles

Fig. 9 Virtualized CoBASA



planning complete, the agent consortiums are generated and saved in the form of nested shop floor profiles. The PM then allocates the workloads in the virtualized environment and the normal hierarchical execution starts.

The Coordinating Agent (CA), as defined in the CoBASA architecture, has the role of generating run time events to the PM module, in order to allow re-configuration of the shop floor control architecture. In practice these events can be:

- *Resource Breakdown* when a virtualized MRA detects that the physical robot has malfunctioned or if the MRA itself crashed, the CA generates a resource breakdown event to the PM. The PM will handle the event by re-creating the shop floor profile representing the virtual consortium in order to replace the robot (if there is a redundant resource available).
- *Rush Order* when the COM module accepts an urgent order from a customer. This rush orders will usually cause the system to move from hierarchical to heterarchical operation mode. The PM will handle this by recreating the shop floor profiles to reflect heterachical operation virtual consortiums based on a default configuration.

The agentification process remains the same as described in the CoBASA architecture. However, the actual agents are running in a shared environment (generic agents and Agent Machine Interfaces or AMIs) as illustrated in Fig. 9.

While the CoBASA MES architecture is very well suited to the virtualization approach provided in this paper, in practice similar mappings between MES components and shop floor profiles can be implemented with almost any decentralized MES implementation. While virtualization itself does not alter or enhance the functionality of the MES implementation used, it does enhance some non functional characteristics, like configuration flexibility, scalability and fault tolerance. A more detailed discussion on the virtualization benefits and limitations is provided in "Virtualization advantages and limitations for intelligent manufacturing systems" section.

Pilot implementation and experimental results

The virtual MES and shop floor concept introduced in this paper was evaluated in the context of a pilot manufacturing system of the CIMR Centre within University Politehnica of Bucharest, presented in Fig. 10. The goal of this pilot implementation is to virtualize an existing holonic MES implementation and analyze the impacts of this on the existing MES implementation. As the MES functionality itself does not change in the virtualized environment, the main focus is to evaluate the virtualization overhead introduced. To do this, an extensive set of message propagation tests are performed, to determine the virtualization overhead in relation to a baseline measurement. These tests help in observing the impact of different workload allocations strategies in the private cloud environment, in the context of the manufacturing system control architecture. The goal is to determine the best strategies to collocate the workloads in a way that would assure acceptable overhead and efficient utilization of the private cloud resources.

Fig. 10 Holonic manufacturing system (CIMR centre)

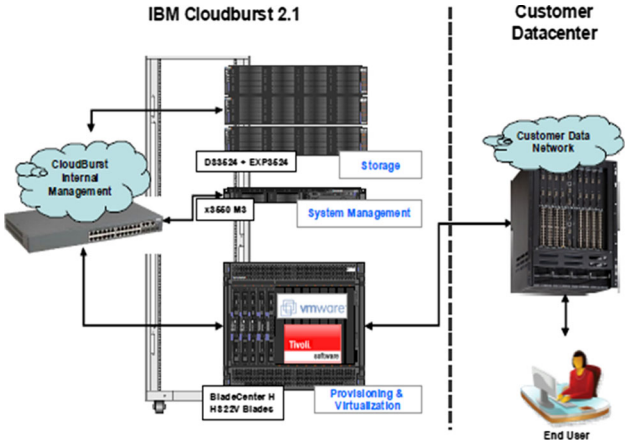


Fig. 11 IBM CloudBurst 2.1 private cloud solution

Shop floor and private cloud resources

On the shop floor there are six robots produced by Adept Technology and a closed-loop twin-track Bosch-Rexroth transportation system with branching and six work places equipped with palette detection sensors and branching capabilities. From a computing perspective, each robot is controlled by a dedicated workstation with the following characteristics: CPU 2.4 GHz, 1.25 GB RAM, 40 GB HDD, 2 Gigabit Ethernet, CD-ROM. The MES implementation is based on a multi-agent system MSB and is executed partially on a dedicated server: xSeries 3500 Server, 4 CPU Xeon QuadCore 2.66 GHz, 24 GB RAM, 8 HDD 7200RPM SATA 500 GB, 12 Gigabit Ethernet, 2 switches 16 ports 10/100.

The product pallets are equipped with Gumstix Overo® Air ARM Cortex-A8 OMAP3503 based computer-on-module, with 600 MHz processor, 512 MB RAM and 802.11g/Bluetooth communication devices.

The private cloud infrastructure is provided by an IBM CloudBurst 2.1 medium size instance (Fig. 11). The IBM Cloud Burst system is an offering based on IBM Blade Center that adds virtualization capabilities using the VMware ESX 4.1 hypervisor, VMware VCenter 4.1 and enhanced administration capability by leveraging the Tivoli Service Management Stack. The virtualization platform is powered by 14 blade servers with two Intel processors with 6 cores each at 2.8 GHz and 12 MB L3 cache. The installed memory is 72 GB for each blade. The estimated capacity is in the area of 400 concurrent virtual machines, considering an optimum scheduling with mixed CPU/IO profiles.

Virtualization process

The virtualization process was performed in several steps. First step consisted in the virtualization of the six PC based workstations controlling the Adapt robots. This process resulted in 6 Windows 7 64bit based VMs with the Adapt V+ software stack installed, each mapped with two virtual network interfaces. One dedicated for management, specifically the integration with the MES/MSB layer and the other for direct Ethernet connection with the Adapt robots on the shop floor. Each VM was allocated with two virtual cores and 2 GB RAM.

The second step involved the virtualization of the shop floor scheduler and production tracking module, residing on the xSeries 3500 Server. Due to the system design the decision was taken to split the workload in two virtual machines, one dealing with the production scheduling in hierarchical mode and one with the heterarchical operating module implemented with multi agent systems in a MSB architecture. This approach decouples the two systems that used to share the same physical server and so improves the overall reliability of the system. The third step involved the virtualization of the conveyor controller workstation on a dedicated Linux based virtual machine with 512 MB RAM. In this pilot implementation, the intelligent product driven by the multi agent system running on the product pallet was not virtualized, due the absence of a hypervisor for the ARM Cortex platform that would allow virtualization of the sensors and mapping through the WIFI network. Finally the IP addresses of each Adapt robot and of the conveyor controller were set as fixed and published in the resource catalogue. The virtualization process resulted in eight individual VM templates, published in the service catalogue of the IBM CloudBurst private cloud.

Message propagation tests

As real time communication represents an important characteristic for MES/shop floor devices (robots, conveyor, intelligent products), a comprehensive set of message propagation tests were performed to evaluate the virtualization overhead

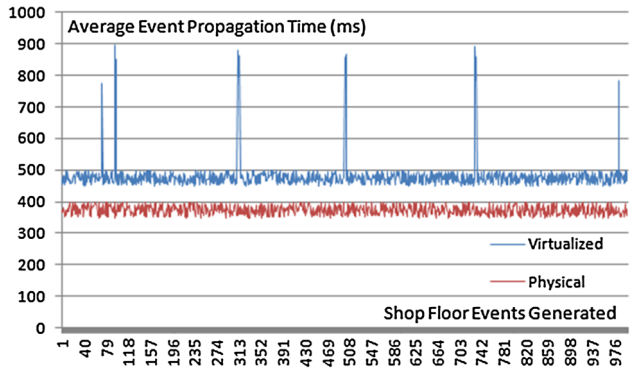


Fig. 12 Average event propagation times (single resource)

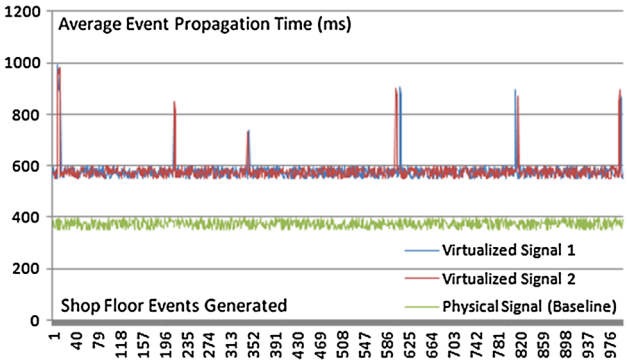


Fig. 13 Average event propagation times (two resources)

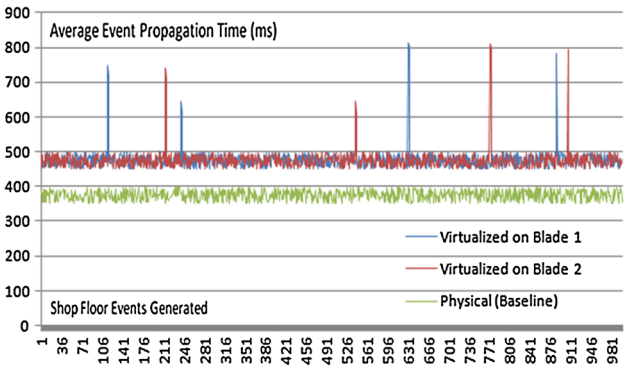


Fig. 14 Average event propagation times (two resources)

introduced. The test scenario consists in generating a set of 1,000 shop floor events one after the other at a predefined and fixed time interval of 500 ms. The event consists in a +5V signal applied to one of the Adapt robot range sensors. The implementation of this integration is using an adapter for wire protocol conversion (see Section IV, Fig. 3b). The signal is propagated in the following order through the controller, network, hypervisor, OS driver and finally to the software agent for the resource, where is logged. The clocks of the signal generator and the virtual machine are synchronized and so the propagation time for the event can be measured. As a baseline, the same propagation time was measured against the physical resource controller. Figure 12 illustrates the propagation time for the 1,000 events in the virtualized workload (blue) and the physical system (red).

We can observe a significant propagation overhead of approximately 20 % on average when the workload is virtualized. Also, we observed a series of peaks in the propagation time for a small amount of messages. These peaks are introduced by the guest OS, due to momentary kernel CPU activity. However, the tests show that the propagation time remains within acceptable limits; in the pilot implementation, acceptable limits are less than 1 second delay. This is the timeout setting for the service based communication between MES agents for resource operations. This timeout is configurable and would vary between specific implementations, depending on how fast the operations are executed on the shop floor.

The second scenario considered two resources generating events and the corresponding agents are collocated on the same virtualized workload. The test results are presented in Fig. 13.

The results show that collocating the agents on the same virtualized workload has an important impact on the propagation overhead, which is in these conditions around 30 % increased compared to the baseline.

Finally, in Fig. 14, is illustrated the situation in which two resources are generating signals and the corresponding agents are isolated on two workloads deployed on independent blades.

In this case the virtualization overhead remains at around 20 %, which proves the scalability of the system if the resource workloads are on isolated blades.

Lessons learned

Based on the experience gathered with the pilot implementation, a set of best practices and implementation advices can be derived:

- *Use a step by step approach* start with the atomic, self contained workloads as there the virtualization is straight forward.
- *Test after each step* the virtualization process should not affect the overall system functionality. It should be a transparent process for the MES/MSB/upper layer systems. This means that after each step tests can be performed to validate the implementation.
- *Workload collocation* a profile of the workload can reveal operational characteristics like CPU intensive, IO intensive, etc. This information can optimize the allocation strategy of the workloads on the virtualization blades.

- *Virtualization overhead* one of the disadvantages of virtualization is the overhead induced on the applications. In real time systems, like manufacturing cells, the virtualization overhead can play an important role, so it must be considered at design time.

Virtualization advantages and limitations for intelligent manufacturing systems

Server virtualization benefits have been outlined and discussed by both research and industry in the last decade. While all server virtualization benefits apply implicitly to manufacturing enterprises, some are specific for this industry in the context of shop floor and MES virtualization. Alongside with the benefits there are a set of limitations of the virtualization approach presented.

Virtualization advantages

Decoupling resource controller virtualization allows a separation between the physical resource and the controlling information system. The most important advantage introduced by decoupling is the possibility to have multiple versions of the virtual controller with different configurations and switch between them as needed. This capability is very useful when initial configuration or changes need to be tested and implemented on the production line. At the same time, based on this, a rollback option is implicitly available.

Redundancy with virtualized resources a redundancy mechanism with active/active or active/passive workloads can be implemented. In traditional implementations of manufacturing cells, implementing a controller redundancy involved using a separate physical workstation, and so it had a high overhead. However, without controller redundancy is impossible to assure high availability for resources. In other words, if the controller workstation faced a problem at runtime, the whole shop floor resource would be unavailable. A private cloud implementation would be able to detect a workload failure and switch to the redundant workload in a very short time, assuring a high uptime for the shop floor resource.

Energy efficiency server virtualization is improving dramatically the energy footprint of the workloads. In most cases, the energy consumption is reduced to 15–20 % of the physical implementations. In the pilot implementation described above, by virtualizing the six workstations and two servers, the estimated energy cost savings per year are $27.000, according to the VMware Green Calculator [VMware (VMware ROI TCO Calculator for Server and Desktop Virtualization, available online, 2013, http://roitco.vmware.com/vmw/)]. These estimates include also the cooling required for the physical workloads.

High performance some operations on the shop floor would require high performance computers for specific tasks. For example, real time image analysis for error detection or for material selection would require high CPU resources. With virtualization and private clouds, such resources can be allocated virtually according to the needs of the specific manufacturing line. A practical example can be the scenario where the manufacturing line is executing at 100 products per hour, with a requirement to process 1,000 images per product. For this, eight virtual processors can be allocated to the image analysis workload. If the manufacturing line will later execute at 50 products per hour, the processing power required will be half, so the workload can be configured with 4 virtual processors. This fine adjustment of the processing power to the actual product manufacturing requirements is not possible without virtualization.

Physical security the data centre environment provides superior physical security for the workloads compared to the shop floor environment. Some of the common approaches is restricting the physical access to the data centre to selected personnel, biometric access control and permanent surveillance.

Shop floor space for manufacturing enterprises, shop floor space is a precious operational resource as it can be used to accommodate parts and raw materials stocks. Virtualization helps free up the space taken by the physical workstation based controllers. In the pilot implementation presented in this paper, 10 cubic meters were made available on the shop floor with the virtualization of the workloads.

Virtualization limitations

Real time operations the main limitation is assuring a virtualized environment can keep up with a real time manufacturing system. Although, the event propagation tests presented in this paper, show promising results for most applications, where shop-floor resources and the manufacturing control system operate in time ranges larger than the propagation overhead seen, this still remains the major limitation. The legacy systems were able to provide guaranteed just-in-time handling to shop floor events, which cannot be completely assured by virtualized environments due to their core nature of shared hardware environments. For some real time manufacturing systems, this still remains the main limitation of adopting virtualization at shop-floor layer.

Development costs moving from the legacy system to a near real-time virtualized implementation adds a series of development costs to the enterprises. Currently, shop floor device manufacturers provide limited or no support for vitalizing their hardware; so the costs of implementing drivers for mapping the physical devices to private cloud workloads and other required adapters fall within the implementing

company area. These costs can become significant for large implementations.

Initial investment the initial investment in the private cloud architecture for shop-floor virtualization considering both the hardware purchase and the associated software can represent an obstacle, especially if the private cloud is dedicated to the shop floor virtualization project. The effectiveness of the private cloud in this context depends on the sharing of the resources among business units in order to achieve optimal resource utilization and to reduce the operational costs.

Training and know how like with any cutting age technology adoption, it has an associated limitation in finding the knowledgeable resources that can implement it and exploit it to the full potential. Training of the engineers designing and implementing such a solution is an important pre-requisite for a successful implementation.

Conclusions and future work

This paper introduces the MES and shop floor virtualization as a solution to reduce operational costs and improve the flexibility, agility and maintainability of the manufacturing system. The shop floor profiles concept introduced in this paper allows quick switching between different versions of workloads and different configurations and bindings of resources. Another important advantage introduced by the shop floor profiles is the possibility to implement test to production (T2P) scenarios, by using revisions and sub-profiles.

Virtualization brings many advantages also on the manufacturing system reliability by allowing full system snapshots and backups and quick recovery in case of failures, as well as providing built in redundancy. Most private cloud implementations offer these features by default and can be directly adopted. Resource controller virtualization allows a separation or decoupling between the physical resource and the information system. The most important advantage introduced by decoupling is the possibility to have multiple versions of the virtual controller with different configurations and switch between them as needed. This capability is very useful when initial configuration or changes need to be tested and implemented on the production line. With virtualized resources a redundancy mechanism with active/active or active/passive workloads can be implemented. In traditional implementations of manufacturing cells, implementing a controller redundancy involves the use of a separate physical workstation, and so it induces a high overhead.

However, without controller redundancy it is impossible to assure high availability for resources. A private cloud implementation would be able to detect a workload failure and switch to the redundant workload in a very short time, assuring a high uptime for the shop floor resource.

Future work is focused on developing an integrated framework for MES and shop floor virtualization based on the concepts presented in this paper.

References

Babiceanu, R., & Chen, F. (2006). Development and applications of holonic manufacturing systems: A survey. *Journal of Intelligent Manufacturing*, *17*, 111–131.

Barata, J., & Camarinha-Matos, L. M. (2003). Coalitions of manufacturing components for shop floor agility—the CoBASA architecture. *International Journal of Networking and Virtual Organizations*, *2*(1), 50–77.

Bellifemine, F., Poggi, A., & Rimassa, G. (2001). JADE: A FIPA2000 compliant agent development environment. In *Proceedings of the 5th international conference on autonomous agents* (pp. 216–217). ACM.

Borangiu, T. (2009). A service-oriented architecture for Holonic manufacturing control. *Springer Series Studies in Computational Intelligence*, *243*, 489–504.

Chappell, D. A. (2009). *Enterprise service bus*. O'Reilly Media Inc.

Cheng, Y., Tao, F., Zhang, L., Zhang, X., Xi, G. H., & Zhao, D. (2010). Study on the utility model and utility equilibrium of resource service transaction in cloud manufacturing. In *Industrial engineering and engineering management (IEEM), IEEE international conference* (pp. 2298–2302).

Gavrilovska, A., Kumar, S., Raj, H., Schwan, K., Gupta, V., Nathuji, R., & Saraiya, P. (2007). High-performance hypervisor architectures: Virtualization in hpc systems. In *Workshop on system-level virtualization for HPC*.

Jamil, D., & Zaki, H. (2011). Security issues in cloud computing and countermeasures. *International Journal of Engineering Science and Technology (IJEST)*, *3*(4), 2672–2676.

Kaufman, L. M. (2009). Data security in the world of cloud computing. *Security & Privacy, IEEE*, *7*(4), 61–64.

Lee, Y. C., & Zomaya, A. Y. (2012). Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing*, *60*(2), 268–280.

Leimeister, S., Böhm, M., Riedl, C., & Krcmar, H. (2010). The business perspective of cloud computing: Actors, roles and value networks. In *18th European conference on information systems, ECIS 2010*.

Leitão, P., Marco, J., Bepperling, A., Cachapa, D., Colombo, A. W., & Restivo, F. (2012). Integration of virtual and real environments for engineering service-oriented manufacturing systems. *Journal of Intelligent Manufacturing*, *23*(6), 2551–2563.

Li, B. H., Zhang, L., Wang, S. L., Tao, F., Cao, J. W., Jiang, X. D., et al. (2010). Cloud manufacturing: A new service-oriented networked manufacturing model. *Computer Integrated Manufacturing Systems*, *16*(1), 1–7.

Liu, N., & Li, X. (2012). A resource virtualization mechanism for cloud manufacturing systems. In M. Sinderen, P. Johnson, X. Xu, & G. Doumeingts (Eds.), *Enterprise interoperability (Lecture Notes in Business Information Processing)* (Vol. 122, pp. 46–59). Berlin: Springer.

Lohr, S. (2007). Google and IBM join in 'cloud computing' research. *New York Times*, *10*(8). http://www.nytimes.com/2007/10/08/technology/08cloud.html?_r=0.

Luo, Y., Zhang, L., Tao, F., Ren, L., Liu, Y., & Zhang, Z. (2013). A modeling and description method of multidimensional information for manufacturing capability in cloud manufacturing system. *International Journal on Advance Manufacturing Technologies, 69*(5–8), 961–975.

Mason-Jones, R., & Towill, D. R. (1997). Information enrichment: Designing the supply chain for competitive advantage. *Supply Chain Management*, *2*(4), 137–148.

McFarlane, D., Sarma, S., Chirn, J. L., Wong, C. Y., & Ashton, K. (2002). The intelligent product in manufacturing control and management. In *Proceedings of 15th Triennial World Congress*. Barcelona, Spain.

Mell, P., & Timothy, G. (2011). The NIST definition of cloud computing (draft). *NIST Special Publication*, *800*(145), 7.

Mergen, M. F., Uhlig, V., Krieger, O., & Xenidis, J. (2006). Virtualization for high-performance computing. *ACM SIGOPS Operating Systems Review*, *40*(2), 8–11.

Meyer, G., Främling, K., & Holmström, J. (2009). Intelligent products: A survey. *Computers in Industry*, *60*(3), 137–148.

Morariu, C., Morariu, O., & Borangiu, T. (2012). Manufacturing service bus integration model for implementing highly flexible and scalable manufacturing systems. In *Proceedings of the 14th IFAC INCOM'12 symposium, PapersOnLine* (Vol. 14. Part 1, pp. 1850–1855).

Morariu, C., & Borangiu, T. (2012). Manufacturing integration framework: A SOA perspective on manufacturing. *Information Control Problems in Manufacturing*, *14*(1), 31–38.

Morariu, C., Morariu, O., & Borangiu, T. (2013). Customer order management in service oriented holonic manufacturing. *Computers in Industry*, *64*(8), 1061–1072.

Pallis, G. (2010). Cloud computing: The new frontier of internet computing. *Internet Computing, IEEE*, *14*(5), 70–73.

Pearson, S., & Benameur, A. (2010). Privacy, security and trust issues arising from cloud computing. In *Cloud computing technology and science (CloudCom), 2010 IEEE second international conference* (pp. 693–702).

Raj, H., & Schwan, K. (2007). High performance and scalable I/O virtualization via self-virtualized devices. In *Proceedings of the 16th international symposium on high performance distributed computing* (pp. 179–188). ACM.

Shi, S., Mo, R., Yang, H., Chang, Z., & Chen, Z. (2007). An implementation of modelling resource in a manufacturing grid for resource sharing. *International Journal of Computer Integrated Manufacturing*, *20*, 169–77.

Svantesson, D., & Clarke, R. (2010). Privacy and consumer risks in cloud computing. *Computer Law & Security Review*, *26*(4), 391–397.

Thomas, A., Trentesaux, D., & Valckenaers, P. (2012). Intelligent distributed production control. *Journal of Intelligent Manufacturing*, *23*(6), 2507–2512.

Vecchiola, C., Pandey, S., & Buyya, R. (2009). High-performance cloud computing: A view of scientific applications, In *Pervasive systems, algorithms, and networks (ISPAN), 10th international symposium* (pp. 4–16).

Wang, X., & Xu, X. W. (2013). ICMS: a cloud-based manufacturing system. In *Cloud manufacturing* (pp. 1–22). London: Springer.

Wang, S., Liu, Z., Sun, Q., Zou, H., Yang, F. (2012). Towards an accurate evaluation of quality of cloud service in service-oriented cloud computing. *Journal of Intelligent Manufacturing*. doi:10.1007/s10845-012-0661-6.

Weissman, C. D., & Bobrowski, S., (2009), The design of the force.com multitenant internet application development platform. In *Proceedings of the 35th SIGMOD international conference on management of data* (pp. 889–896).

Witsch, M., & Vogel-Heuser, B. (2012). Towards a formal specification framework for manufacturing execution systems. *Industrial Informatics, IEEE Transactions on*, *8*(2), 311–320.

Wu, L., & Yang, C. (2010). A solution of manufacturing resources sharing in cloud computing environment. In *Cooperative design, visualization and engineering* (pp. 247–252). Berlin: Springer.

Xu, X. (2012). From cloud computing to cloud manufacturing. *Robotics and Computer-Integrated Manufacturing*, *28*(1), 75–86.

Zhang, L., Yongliang, L., Fei, T., Bo, H. L., Lei, R., Xuesong, Z., Hua, G., Ying, C., Anrui, H., & Yongkui L. (2012a). Cloud manufacturing: A new manufacturing paradigm. *Enterprise Information Systems ahead-of-print, 8*(2), 1–21.

Zhang, Y., Pingyu, J., Huang, G., Qu, T., Zhou, G., & Hong, J. (2012b). RFID-enabled real-time manufacturing information tracking infrastructure for extended enterprises. *Journal of Intelligent Manufacturing*, *23*(6), 2357–2366.