# Research on the Architecture and its implementation for Instrumentation and Measurement Cloud

Hengjing He, Wei Zhao, Songling Huang, Geoffrey C. Fox

**Abstract:** Cloud computing has brought a new way for resource utilization and management. Nowadays some researchers are working on cloud based instrumentation and measurement system. However, till now, no detailed architecture with implemented system are presented and most of the research work is about applications of cloud computing technologies in instrumentation and measurement field. This paper adopts the philosophy of cloud computing and brings forward a novel architecture for instrumentation and measurement cloud. This architecture has many key features of cloud computing, such as service provision on demand, scalability, load balance and so on. And these features are for remote instrumentation and measurement system. In the architecture, instruments and sensors are virtualized and instrumentation and measurement function modules are wrapped into services. Users can utilize these resources and services on demand remotely. Such architecture can greatly reduce the investment for building remote measurement system, increase utilization efficiency of resources and provide effective platform for processing and analyzing big data from instruments and sensors. System with typical application are implemented upon the architecture. Results of the test show that the new architecture has achieved the design goals and demonstrate the novel instrumentation cloud architecture is an effective and efficient platform for building remote instrumentation and measurement systems.

**0 Introduction**

Since instrumentation and measurement(IM) technology is closely combined with information technology, development in information technology(IT) can lead to the advance of IM technology. In the early stage, computers were introduced to control instruments and sensors for local data acquisition and analysis. Later on, virtual instrumentation technology was developed and many of the functions that were implemented by hardware in instrument can now be achieved by computer software. With the development of networks and internet, remote instrumentation and measurement(RIM) emerged as a new technology in IM[[1](#_ENREF_1)]. Such RIM technology has brought lots of benefits to related areas, especially to those areas that involve large distributed systems[[2](#_ENREF_2)]. It can greatly facilitate instrument control, data acquisition and process. Additionally, new computing paradigms, such as grid computing, are integrated into IM technology to further improve the ability of data process and resource sharing and management for distributed IM systems[[3](#_ENREF_3)]. Grid enabled instrumentation and measurement system(GIMS) is a representative of those systems. GIMS brings many advantages to data intensive IM applications and heterogeneous IM resource management. However, due to some limitations of grid computing, systems that integrating grid are eventually not widely adopted in practical use.

Currently, most of IM systems are built upon local architecture or traditional client/server(C/S) architecture[[4](#_ENREF_4)]. In local IM architecture, instruments and sensors are connected directly to the computer and it is difficult to build large IM systems. As for C/S architecture, instruments and sensors simply provide remote access interfaces through gateway servers to clients. However, both of the architectures require the user to build the entire IT system and, also, to maintain all resources. Thus, there has to be a great investment in building the whole IM system and, besides, system stability, scalability and fault tolerance can be a big problem for both of the architectures. Moreover, to satisfy resource requirement for both peak and valley load, the IM system should be built according to the peak load at the very beginning and, even if the load drops, the system cannot scale down accordingly. Therefore, the utilization rate of resource in the IM system can be low, especially for those systems with great load dynamics. In addition to the problems mentioned above, large amount of data collected from various devices need much more powerful computing resource for processing and analyzing, and traditional computing paradigms may be incapable of dealing with such scenario.

In recent years, the emergence of Cloud Computing has brought many new approaches for resource utilization and management[[5](#_ENREF_5)]. Cloud manages all resources as a resource pool and provides those as online services to end users according to their demand. Such mode can greatly increases the utilization efficiency of resources and, at the same time, save the investment of users on both hardware and software IT resources[[6](#_ENREF_6)]. Also, big data processing and analyzing technologies developed along cloud computing make data analyzing much easier and faster in IM applications[[7](#_ENREF_7)]. Motivated by these benefits, many researchers are exploring novel cloud based IM technologies to solve the problems above[[8](#_ENREF_8)]. Till now, most of the work carried out in the interdisciplinary area of IM and cloud computing mainly focuses on application of cloud computing in IM systems, which can only deal with a few aspects of related problems. Little research is carried out to build novel IM modes and architectures that can inherit the essence of cloud computing for IM systems. Some literatures have brought up new concepts or terminologies such as instrumentation cloud or sensor cloud, but with only conceptual architectures. However, design such an instrumentation and measurement cloud with detailed architecture and corresponding practical system is very important for the current IM science to face the many aforementioned challenges.

This paper introduced a novel IM cloud architecture with detailed system implementations. The architecture abstracts instruments and sensors into resources, and frequently used modules and functions into services. Services are deployed in the cloud and users can consume these services on demand. IM applications are also deployed and running in the IM cloud. All IT resources are allocated and managed by IAAS(Infrastructure as A Service) cloud platform which will reduce investments for users and also increase resource utilization efficiency. By integrating cloud computing and big data processing technologies, the IM cloud can benefit a lot from advantages such as system scalability, fault tolerant, distributed and parallel computing, and so on. An actual system based on this architecture is implemented using various cloud computing frameworks. Applications and experiments are designed to test the system. Results show that the IM cloud architecture designed in this paper can properly integrate cloud computing with IM technologies and greatly facilitate building, managing and using of IM systems and resources.

The remainder of this paper is organized as follows: section II presents related work; section III introduces major concepts of the IM cloud; section IV describes detailed IM cloud architecture designed by this paper; section V illustrates the implementation of the architecture; section VI provides some applications and tests over the IM cloud system; finally, section VII concludes the whole paper.

**1 Related work**

Most of the work related to IM cloud mainly focuses on the following areas: Grid-enabled instrumentation system(GEIS)[[9](#_ENREF_9)], sensor cloud[[10](#_ENREF_10)] and instrumentation cloud[[11](#_ENREF_11)]. GEIS mainly focuses on converting instruments into grid services[[12](#_ENREF_12)], so that heterogeneous instrumentation resources can be accessed and managed through grid, which can facilitate data intensive applications to use various grid resources and provide distributed instrumentation and experiment collaborations over the grid[[13](#_ENREF_13)]. Instruments are abstracted into unified services through middleware technologies and standard modeling. Typical GEISs including Instrument Element[[14](#_ENREF_14)] architecture from GridCC project, common instrument middleware architecture[[15](#_ENREF_15)], e-infrastructure for remote instrumentation from DORII project[[3](#_ENREF_3)] and virtual laboratory architecture[[16](#_ENREF_16)]. Although GEISs provide a good way for distributed instrumentation over gird for collaborative experiments, limitations of grid computing prevented them from prevail among scientific and industrial areas.

Just like IM cloud, sensor cloud is also a very new concept. In[[17](#_ENREF_17)] a sensor cloud infrastructure is developed and physical sensors are abstracted into virtual sensor objects. Such virtual sensor objects combined with related sensor definition template can provide measurement services to end users. Besides, service and accounting models are designed, which makes the sensor cloud infrastructure conforms the philosophy of cloud computing. Some other researches also use the terminology sensor cloud, but most of them only concentrate on the application of cloud computing technology in sensor control and management[[18-21](#_ENREF_18)]. Sensors are much simpler than instruments, however they can also be treated as instruments but only with less functions.

Current studies of instrumentation cloud or similar research areas only bring forward conceptual models and architectures with few details or implementations provided[[11](#_ENREF_11), [22](#_ENREF_22), [23](#_ENREF_23)]. Other research work is mainly about the applications of cloud computing in massive data storage and processing[[24-26](#_ENREF_24)], which, as explained before, only provides solutions for a few problems faced by current IM systems.

**2 Instrumentation and Measurement Cloud**

The key ideas of cloud computing are: resource abstraction and virtualization, services delivered on demand, and scalability[[27](#_ENREF_27)]. Own to these ideas, cloud computing can provide scalable, on demand hardware and software resources remotely. To inherit these important advantages, the IM cloud should also follow these ideas. First of all, instruments and sensors need to be abstracted into virtual resources so that they can be accessed and managed remotely. Unlike computing, storage and networking resources, instruments and sensors have heterogeneous software and hardware architecture which means it is very difficult to virtualize them into unified resource pool and at the same time keep their own features. However, thanks to the virtual instrumentation(VI) technology and sensor modeling standard, many of the modern instruments and sensors can provide unified access interfaces. But such interfaces are designed just for local drivers, to access them remotely interface remapping through networks is still required. Generally, there are two ways for remapping the interfaces as shown in Fig. 1.



1. (B)

Fig. 1 Remote instrument and sensor interface remapping

Fig. 1(A) shows an interface remapping scheme using Remote procedure call. This method is easier but has less flexibility. For different VI frameworks or sensor models, corresponding RPC module should be developed. The second method remotely maps physical interfaces, such as USB[[28](#_ENREF_28)], RS-232, GPIB and many other, of a device to the cloud side. Physical interface remapping actually remaps the device to the remote server or applications and, thus, it supports full features of the device. However this method is much more difficult than using RPC, and also this method only support VM based IM applications, which means each resource should be attached to a VM in the cloud. IM system based on physical interface remapping are more like a traditional IM system deployed in the cloud. Unless there are special requirements, it is better to use RPC framework for IM resource virtualization.

In the context of this paper, service in IM cloud is defined as a running IM function module that can be shared by multiply IM applications. The service model of IM cloud is more close to the concept of SAAS(Software as a Service) in cloud computing[[29](#_ENREF_29)]. When designing IM applications, users just need to know the interfaces of the service, send data through input interfaces to the service module and retrieve results from output interfaces. Fig. 2 demonstrates the scenario of using services in IM cloud.



Fig. 2 Services in an IM cloud

Fig. 2 shows services for real-time IM stream data processing. Here real-time means data are processed as soon as they are received and it is distinguished from batch processing. Since data from virtualized IM resources are normally stream data, they should be processed in time before storing to databases or files. In an IM cloud, services are running in the cloud and each type of service can serve multiple IM applications by running multiple instances. In this way, resources can be shared between users and, as commonly used modules are encapsulated into services, users do not need to develop those modules again, which will save a lot of work in programming. By consuming IM cloud services and deploying applications in the IM cloud, users can save lots of investments on IT facilities and related software and hardware maintain work when building a remote IM system. Also, services are provided on demand and users can consume those services according to their own needs, which will further reduce cost.

Scalability in an IM cloud is achieved by scaling both cloud computing resources, such as VMs, storage and networking, and service instances. As the number of devices and applications in an IM cloud is very large, the feasible way to handle these entities is to utilize parallel and distribute computing frameworks from big data analysis technologies. These frameworks normally provide dynamic scaling and load balancing interfaces. When the load on a service module increases, the scheduler module of an IM cloud can launch new service instances, or add VMs or physical nodes to the framework that runs this service and rebalance the load. Scalability along with on demand service provision guarantee the high resource utilization efficiency.

Although technologies from IM science, cloud computing and big data field provide full support for the above basic requirements of an IM cloud, there was no detailed architecture to integrate all these technologies and guide the implementation of an IM cloud. In the following section, a novel architecture for the IM cloud that has the above important characteristics will be presented.

**3 Novel architecture for Instrumentation and Measurement Cloud**

The overall architecture for IM Cloud developed in this paper is shown in Fig 3. This architecture mainly consist of four parts, which are IM cloud resource agent, IM cloud service pool, IM cloud application executor and IM cloud manager.



Fig. 3 Novel architecture for the IM cloud

The IM cloud resource agent is responsible for instrument and sensor virtualization and IM resource register. As illustrated in the previous section, instruments and sensors can be virtualized through VI frameworks and sensor modeling. By virtualization, instruments and sensors are encapsulated into resources with standard access interfaces and, assisted by RPC frameworks, these interfaces can be called remotely from IM applications in the cloud, which will bring much convenience for building distributed IM systems. Once virtualized, these IM resources can be registered into IM cloud by IM cloud resource agent so that users can use them over networks. To use IM resources, users should first make a reservation for each resource through IM cloud manager. After reservation, an access id with start and end time stamps will be allocated to user applications and IM cloud resource agent, and also an resource entry, such as an IM resource agent URL, will be sent to user application. When user applications need to access to the reserved resources, they will have to send access id to IM cloud resource agent through that entry first and the IM cloud resource agent will then check if it also has the same access id and whether current time is between the reserved time span. If all requirements are satisfied, IM cloud resource agent will allocate a resource handler for the application and allow the application to use the resource through RPC interfaces for instrumentation and measurement tasks. The complete procedure of utilizing IM resources is shown in Fig. 4.

In IM cloud, services represent modularized function blocks implemented in big data analyzing and cloud computing frameworks. Such IM cloud services include stream data processing modules, batch processing modules and other modules. IM cloud services are normally developed and deployed in parallel and distributed cloud computing platforms. Each type of service can serve multiple IM applications and the platform or framework running these services will provide scaling, parallel processing and fault tolerance abilities. Services and applications are connected by Pub/Sub based message bus. Message bus is used to transmit data between services, resources and applications. For stream data processing services, each of them has several input and output interfaces. Input interfaces will listen on dedicated message topics and, as for output interfaces, they will watch on a data cache entry that stores destination message topics of output data. Fig. 5 shows the typical service model for processing steam data from IM cloud applications.



Fig. 4 Steps required for utilizing IM resources



Fig. 5 Steam data processing service model in the IM cloud

All services in the IM cloud should register themselves through IM cloud manager. When registering, the IM cloud manager will create data entry for each interface in the coordination system and write Meta data of interfaces into those entries. For input interfaces, the IM cloud manager will write topics that they are listening to their data entries and if IM applications are to consume this service, they can get those topics through the IM cloud manager and then publish data to those topics. To get processed data streams from the service, IM applications need to write topics that their sink models are listening to data entries of the service output interfaces.

For batch processing services, file systems and databases constitute the data sources. Since, in most cases, batch processing is an off-line post-processing approach but IM systems normally deal with real-time stream data processing, it will not be studied in this paper. However batch processing services are still indispensable to the whole IM cloud architecture.

To enable parallel and distribute computing paradigm and enhance the ability of fault tolerance for IM cloud services, three key roles should be followed when developing IM cloud services:

1. Reduce coupling and dependency between data. Only in this way can the service utilize parallel computing.

2. Make data self-descriptive. This is very important for multiple IM cloud applications to share the same service instance. Since data can describe themselves, service instance does not need to know which application these data come from.

3. Try to avoid state caching for IM cloud applications in services and use dedicated memory cache systems. Most of distributed and parallel cloud computing frameworks support fault tolerance. That means when some processing nodes go down the system can still run in normal state. However, if those nodes cache states of current applications they serve, all these states will be lost and restoring the service process can be difficult, especially for streaming processing applications. Moreover, avoid state caching in service instances can facilitate online load transfer which is vital to load balancing.

The IM cloud application executor is responsible for running IM cloud applications that are deployed in the IM cloud. It often consists of script interpreter or language runtime engine. By deploying IM applications into the IM cloud, users can save the trouble of maintaining the client side. With proper user interfaces, users can access to their IM applications through mobile terminals.

The kernel of the whole architecture is the IM cloud manager. IM cloud manager contains four main components: the resource manager, the service manager, the application manager and the scheduler. All IM cloud resources, related IM cloud services and IM cloud applications are registered and managed by the corresponding component of the IM cloud manager. Fig. 6 shows how the IM cloud manager works.

When registering resources, resource manager will create a data entry for each resource and store their Meta data. Under RPC framework, Meta data often contains URL of the resource side RPC server. Another data entry that caches all reservation information of the resource is also created upon registration. Such reservation information will also be sent to the IM cloud resource agent for authorization purpose. Similar process happens when service manager registering services. However service manager mainly maintains Meta data about interfaces of each service. Service and resource requests from IM cloud applications are processed by the application manager. When IM cloud applications request services or resources, the application manager will query the coordination system, get all related Meta data and send these data back to applications. At the same time, some of the message bus context of the sink module in the applications will be written into data entries that output interfaces of services are listening on.



Fig. 6 Details of the IM cloud manager

The tasks of the scheduler include IM cloud resource scheduling and load balancing and scaling of IM cloud services. Fig. 7 shows how those tasks are handled by the scheduler.



Fig. 7 Work procedure of the scheduler

When an IM cloud application request for IM cloud resources, the scheduler will call related algorithms to calculate valid time span for those resources and make reservations. Currently most of IM resources are not like services, and they cannot be shared, at the same time, by different users. Thus, scheduling running time of IM cloud resources according to users’ needs is very important. Load balancing and service scaling is done by scaling the cluster that runs the service. When IM cloud applications request services, the scheduler will query the load capacity of each service instance and select the ones with low load capacities. Also when several service instances have low load capacities scheduler can transfer tasks on these instances to on instance and shutdown other instances. Such online load balancing can greatly increase the utilization efficiency of computing, storage and other resources in the cloud. However, online load balancing needs the support from both the framework that run the service and the service itself.

Whenever an IM cloud application is registered in the IM cloud, the application manager will record all consumed services and resources. If the state of any of the services or resources changed, related event will be sent to the application manager and trigger the state transition of the IM cloud application. Also change of application state can trigger state transition of resource. State transition models for service, resource and application in an IM cloud are shown in Fig. 8.



Fig. 8 State transition models of service, resource and application in an IM cloud

All state transition models in Fig. 8 start from unregistered state and this state normally means related entities are not yet connected to the IM cloud. As for resource, if the application utilizing this resource become invalid, resource state will transfer from reserved or activated to registered state, which means this resource is released back to the resource pool. State transition model for service is much simpler since service instance can be shared by many applications at the same time which will save the “reserved state” from scheduling. Application state transition model is a little more complicated as it involves events from resources and services. Invalidation of resources or services will always cause the application to be unregistered, thus, when implementing the IM cloud architecture, only very serious errors or problems are allowed to generate the “invalid” event and in other situations try to pending the application rather than unregister it. Fig. 8 shows only very basic state transition models for IM cloud, however implementation of the IM cloud architecture will need more detailed models to handle more complicated situations.

To demonstrate the feasibility and advantages of the IM cloud architecture brought forward by this paper, practical system implemented upon this architecture is presented.

**4 Implementation of the IM cloud architecture**

First, to verify that the architecture is viable for instrument and sensor virtualization, VISA(Virtual Instrument Software Architecture) based IM cloud resource agent is implemented. The agent uses Apache Thrift RPC framework and all VISA driver interfaces are remotely mapped. Since Thrift supports cross language services development, it is also used as the service framework between all servers and clients in the IM cloud. Fig. 9 shows the details of implemented IM cloud resource agent and how it interacts with the resource manager in the cloud.



Fig. 9 Implementation of the IM cloud resource agent

As shown in Fig. 9, instrument resources are registered through resource management RPC services which are implemented in the Thrift framework. To access the instrument resource, each IM cloud resource agent needs to run a VISA RPC server and it wraps all VISA driver interfaces. However, interfaces are extended to include a third parameter which is the access id. Such access id contains both the name of the instrument resource and the reserved time span of the resource. The resource agent will also store a set of <access id, instrument resource> map and this map is built up when applications in the cloud request the resources managed by this agent. Once the application in the IM cloud needs to control a resource, it will make a remote call with the access id. On the agent side, the agent will get the corresponding resource through this id and then call local VISA instrument driver to control the instrument and then return the result to the application. To make the agent as independent with platforms as possible, pyvisa, which is a python implemented frontend of VISA driver, is used as the VISA resource manager. Although instruments and sensors are diverse in there hardware architectures and software interfaces, similar implementation can be applied to virtualize them into IM cloud resource as long as their interfaces are available.

Coordination system is used to record important data and information about resources, services and applications in the IM cloud. Here, Zookeeper, which is a distributed coordination system with fault tolerant ability, is utilized to store various data. Zookeeper uses a data structure called a Zookeeper path which is similar to the directory of a file system and each node of a path can store data. Various message brokers can be used to implement the message bus in the IM cloud and in this paper Rabbitmq is adopted.

As for resources, the data structure used to record their information is shown in Fig. 10. When registering resources, the resource manager of the IM cloud manager will create a data path according to the domain, site and resource name. Here domain and site are used to constitute a two level naming convention so that resource management can be more convenient and flexible. Whenever a reservation is made for a resource, a duration Id with start and end time will be added as a child to the resource data path when the reservation become valid. And as long as a resource is registered, the IM cloud resource agent will listen on the data path for child adding event. If such event happens, the agent will activated.



Fig. 10 Data paths for IM cloud resource

Then IM services should be implemented. As explained before, IM applications normally need to process real-time steam data, thus stream data processing engine is required and related IM function modules need to be developed. In the work of this paper, Apache Storm[[30](#_ENREF_30)] is used as the real-time streaming data processing engine. Storm is a distribute parallel stream data processing engine with fault tolerance and its maximum processing speed can reach around 1 million messages per second. Storm contains two types of components, which are Spout and Bolt. Spout is the data source and it can be connected to message broker to subscribe message topics. Bolt is the process component. Spouts and Bolts compose a Topology which carry out the stream data processing logic. Each Spout or Bolt can run multiple instances so that they can process data in parallel. Topologies are wrapped into services with certain Spouts functioning as input interfaces and some Bolts as output interfaces. Fig. 11 presents how services in the IM cloud interact with applications and the IM cloud manager in the cloud.

Input interfaces of a service are Spouts which are extended with service management clients. When a service is submitted, these spouts will register their message topic contexts through those clients. To consume a service, an IM cloud application just need to get the context of each input interfaces and send data to them. Output interfaces are also extended Bolts with service management clients and they will also create data paths and listening on those paths. However it is the application manager that is responsible for writing message topic contexts to the paths. Whenever data on paths are updated, output interfaces of a service will update destination contexts and send output data to the corresponding message topic. Each context in an output interface is combined with a session Id and a module Id which will be introduced in the following part, and each message passed to an output interface will also contain these Ids. With these Ids, output interfaces will know which message topic the data should be sent to. Data paths for services are shown in Fig. 12.



Fig. 11 A service implemented through Storm to compute power



Fig. 12 Data paths for IM cloud services

Children under in node are input interfaces of the service and similarly children under out node are output interfaces. Transport represents various message brokers. The session node and its children are used for data routing which will be illustrated later. The capacity node stores load capacity of this service and the IM cloud manager will use this data for load balance and scaling.

IM cloud applications are implemented through text programing language and currently only Java APIs(Application Programming Interface) are developed. Four classes are defined according to entities in the IM cloud which are ICResource, ICService, StreamProducer and StreamConsumer. ICResource and ICService are wrappers of the resources and services in the IM cloud. When creating ICResource objects, domain, site, resource name and time span should be specified, but for ICService objects only service name is required. ICResource class is normally wrapped with RPC interfaces that is used to control resources. StreamProducer is used to publish data to the input interfaces to a service while StreamConsumer is responsible for receiving data from service. However, all related message broker contexts are automatically set when submitting the application. A complete IM process is wrapped into a session task and all related resources, services and other modules are managed through a Session object. All the four types of components in an IM cloud application session have their module Ids and each session has a universally unique Id, but module Id is only unique to a session. Fig. 13 shows a diagram of a simple IM cloud application just for illustration.



Fig. 13 An IM cloud application

Code fragment for the IM cloud application in Fig. 13 is as follows.

ICResource voltMeter = session.newICResource(“test”,“site1”, “ASRL3::INSTR”, “VoltMeter”);

ICResource currentMeter=session.newICResource(“test”,“site1”, “TCPIP0::localhost:2222::inst0::INSTR”, “CurrentMeter”);

StreamConsumer consumer = session.newSink(“Sink”);

StreamProducer producer = session.newHeader(“Source”).consumeService(“Power\_m”, “Input”);

session.newICServiceModule(“Power\_m”, “Power”).consumeService(“Output”, “Sink”);

There is a consumeService method that is used to decide which module and interface the data of current module should be sent to. For StreamProucer module, the name of the next module and the interface should be provided while for service module the name of the service output interface and the next module and its input interface, if any, should be defined.

Data routing in IM cloud is relatively more complex than conventional IM programs. Fig. 14 shows how data are routed between multiple services.



Fig. 14 Data routing in the IM cloud

In Fig. 14 two services are consumed by the application. Here simple Ids are used just for the convenience of illustration. Data published by StreamProducer will be filled with session Id and module Id of the consumed service and these two ids will be passed all though the service until reaching the output Interface. Session Id and module Id of the service will also be stored under the session node in the coordination system. This data entry is used to make sure that multiple inputs from the application are routed to the same service instance. For example, in Fig. 14 voltage and current value should be sent to the same service instance for computing power. When submitting the application, the application manager will search all instances of service Power for the session Id and service module Id when processing service request for voltage StreamProducer modV. If no session Id named SessionPower is found, the application manager will choose the service instance with lowest load capacity and write session id and service module id to that instance and then record message broker context of the input interface SpoutV for returning. Similar process happens when processing modI but this time the application manager will choose the instance that has already be written with the session Id. In this way voltage and current from modV and modI respectively are guaranteed to be sent to the same service instance. Once data are processed and sent to the output Bolt, the output Bolt will check the destinations retrieved from coordination system. Destinations store session Ids, current module Ids and the next module Ids and related message broker contexts. With session Id and current module Id, output Bolt will find the context for transferring data, at the same time the module Id will be replaced by the next module Id so that data can be routed similarly in the next service module. Session Id will not be changed all through the session task.

**5 Application**

To test the IM cloud architecture an application for power system state estimation[[31](#_ENREF_31)] is developed based on the implemented system. In power system, not all states of the system are monitored and even some electric states are measured by instrumentation meters or sensors, the value may not reflect the true state due to measurement errors or bad data caused by disturbance to communication systems and many other influences. However all real states of a power system should always obey the law of electric circuit. Thus, using these electric circuit laws and measured data a group of measurement equations can be built up. With these equations a mathematic model can be developed to estimate states of the whole power system. Nevertheless, such estimating process is computation intensive and also consumes lots of memory especially when the system is large, but it is required that the estimation can be as fast as possible. Fortunately the IM cloud can provide parallel processing framework, therefore system can be split into subareas and estimate each area in parallel. Here the estimation module is implemented into an IM cloud service which is a storm topology as shown in Fig. 15.



Fig. 15 Service for power system estimation

The topology contains three parts which are measurement, system splitting and estimation part. The estimation part is further divided into estimation iteration loop and the bad data recognition loop. Following the rules for developing services, all data required for estimation are stored in the Redis database which is an in-memory database.

As for measurement system, pyvisa-sim is used to simulate meters that measures state of a system. The measurement data are coming from power flow of the tested power system. In this paper, case data from Matpower[[32](#_ENREF_32)] is used. Deployment of the whole system is shown in Fig.16 and the whole system is running on an Openstack IAAS cloud platform.



Fig. 16 Deployment of the whole system

Two scenarios are set for testing the IM cloud and all cases in testing are split into subsystem with number of buses less than 300. The first test scenario is running estimation on the IM cloud just for the largest case case9241pegase with 9241 buses. The most computation intensive Bolt are the Estimate Once Bolt and Bad Data Recognition Bolt, thus multiple workers are set for the instances of these two Bolts. To find out which Bolt plays more important role in determining the total estimation time, different number of workers are set for each of the Bolts, estimation time is measured as shown in Fig. 17.

In Fig. 17 PB is the number of workers for Bad Data Recognition Bolt and PE is the number of workers for Estimate Once Bolt. Fig. 17 shows that with the increase of PB estimation time will reduce dramatically. From the figures in the first row of Fig. 17, it can be seen that changing the number of PE does not effectively influence the estimating time while figures in the second row of Fig. 17 demonstrates that increase PB will greatly reduce estimating time. However when PB increases to 12 or larger, estimating time does not decrease any more. This is caused by the overhead from distributing computing, such as data transmitting delay between computing nodes. Fig. 17 demonstrates that bad recognition process is much more time-consuming than estimation process. Fig. 17 also shows that sometimes increasing PB or PE may cause performance degradation that is because instances of Bolt may run on different nodes or on the same node. When more instances running on the same node communication delay can be reduced, however, if instances are running on distributed nodes, overhead from communication delay will degrade overall performance. Currently, distribution of Bolt instances are carried out automatically by Storm and that is the reason for performance fluctuation in Fig. 17. Similar phenomenon will also happen in the following tests.

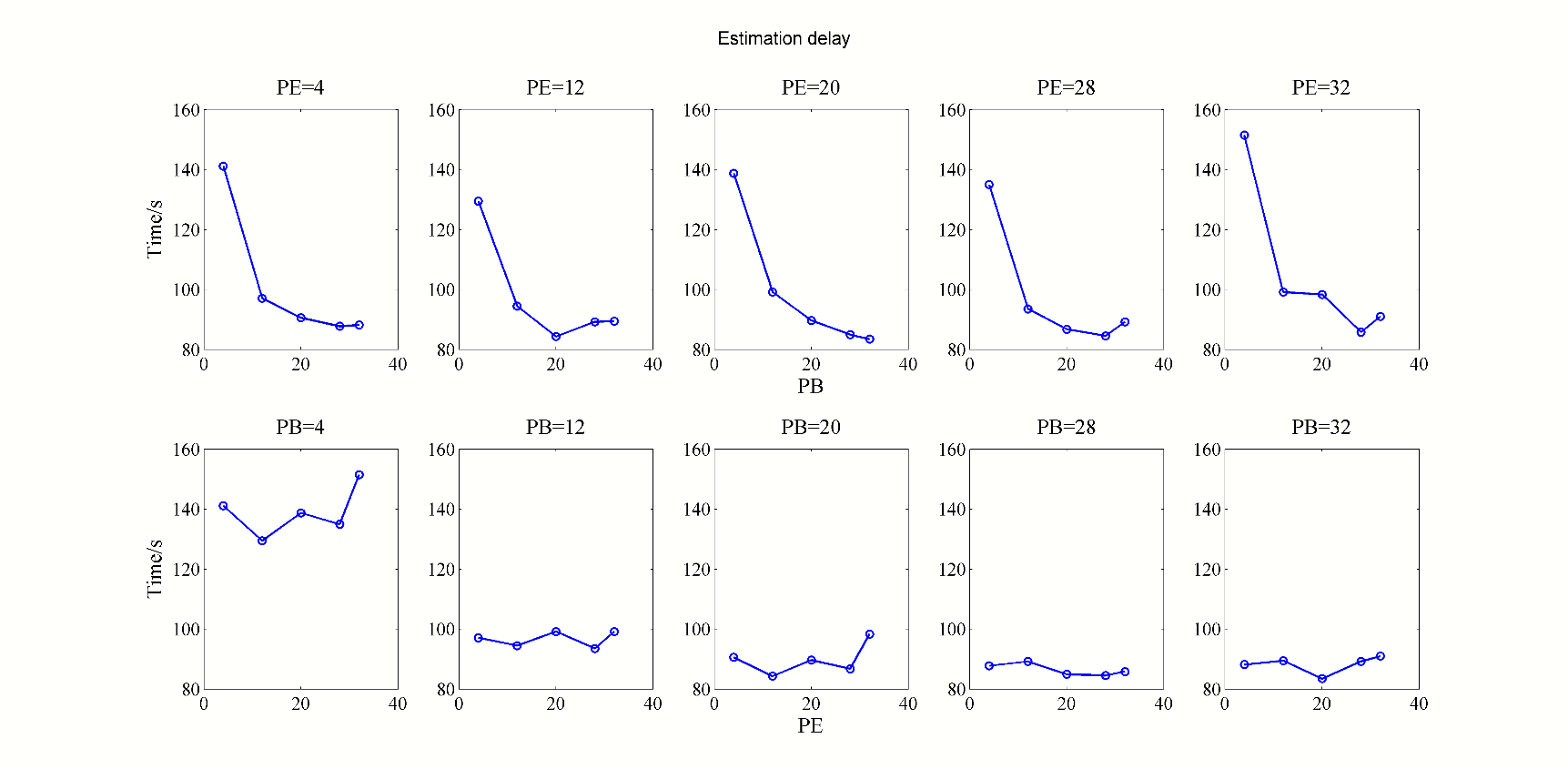


Fig. 17 State estimation time for case9241pegase under different number of workers

The second test scenario runs state estimation for three medium scale cases, which are case2869pegase, case3012wp and case3120sp, simultaneously. As PE does not affect performance much, it is set to 28 and will not change in this test. Fig. 18 shows estimation delays under different number of PB for those three cases.



Fig. 18 State estimation time for multiply cases on the IM cloud

Figures in the up row of Fig. 18 show the time required for one time bad data recognition. Obviously increasing PB can reduce recognition time for all cases. Figures in the down row of Fig. 18 show the estimation time of the three cases. Similarly increase workers can greatly reduce bad data recognition time.

Both Fig. 17 and Fig. 18 demonstrate that the IM cloud can provide parallel processing ability and system performance can be increased by scaling up computing nodes in the Storm cluster. Fig. 18 illustrate that the IM cloud can provide shared IM service to multiple applications.

**6 Conclusion**

The instrumentation and measurement cloud can greatly facilitate management of instruments and sensors and at the same time allows users to utilize those resources and related IM services on demand remotely.

The IM cloud architecture brought forward in this paper provides efficient guidance for developing a practical IM cloud. With IM device virtualization and service wrapping, building a remote IM system just requires only very simple coding work. Most of the investment in IT facilities and system development work can be saved. Also with the ability to scale and load balance, the IM cloud can increase the utilization efficiency of various resources to a much high level. Distribute parallel computing paradigm of the IM cloud will accelerate the processing speed which brings lots of benefits for large scale remote IM system and also for analysis of big data coming from huge number of instruments and sensors. Application developed upon the implemented system has demonstrated the advantages of the work done in this paper.

However, more research work is still required to deal with some challenges. Such challenges include latency and stability of network, geographic characteristics of the physical object that is being measured and so on. These challenges are not proprietary to the IM cloud but common in remote instrumentation and measurement field. But with more effort, problems caused by these challenges can eventually be solved.

**7 Acknowledgement**

This paper is funded by the State Key Laboratory of Power System award SKLD15M02, Department of Electrical Engineering, Tsinghua University, Beijing, China.

**References**

[1] M. Bertocco, Architectures For Remote Measurement, in Distributed Cooperative Laboratories: Networking, Instrumentation, and Measurements, F. Davoli, S. Palazzo,S. Zappatore, Eds.: Springer US, **(2006)**, pp. 349-362.

[2] A. Roversi, A. Conti, D. Dardari, O. Andrisano, A WEB-Based Architecture Enabling Cooperative Telemeasurements, in Distributed Cooperative Laboratories: Networking, Instrumentation, and Measurements, F. Davoli, S. Palazzo,S. Zappatore, Eds.: Springer US, **(2006)**, pp. 395-407.

[3] A. Cheptsov, B. Koller, D. Adami, F. Davoli, S. Mueller, N. Meyer, P. Lazzari, S. Salon, J. Watzl, M. Schiffers, and D. Kranzlmueller, E-Infrastructure for Remote Instrumentation, Computer Standards & Interfaces. 34(2012), **(2012)**, pp. 476-484.

[4] S. Qin, X. Liu, L. Bo. Study on the Networked Virtual Instrument and Its Application. Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2005. IDAACS 2005. IEEE. **(2005)**5-7 Sept. 2005,

[5] Y. Jadeja, K. Modi. Cloud computing - concepts, architecture and challenges. Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on. **(2012)**21-22 March 2012,

[6] I. Foster, Z. Yong, I. Raicu, L. Shiyong. Cloud Computing and Grid Computing 360-Degree Compared. Grid Computing Environments Workshop. **(2008)**November 12-16,

[7] M. Fazio, M. Paone, A. Puliafito, M. Villari. Huge amount of heterogeneous sensed data needs the cloud. Systems, Signals and Devices (SSD), 2012 9th International Multi-Conference on. **(2012)**20-23 March 2012,

[8] W. Rekik, M. Mhiri, M. Khemakhem. A smart cloud repository for online instrument. 2012 International Conference on Education and e-Learning Innovations (ICEELI). **(2012)**1-3 July 2012,

[9] A. S. McGough, D. J. Colling. The GRIDCC Project. First International Conference on Communication System Software and Middleware. **(2006)**,

[10] M. Yuriyama, T. Kushida. Sensor-Cloud Infrastructure - Physical Sensor Management with Virtualized Sensors on Cloud Computing. 13th International Conference on Network-Based Information Systems (NBiS). **(2010)**September 14-16,

[11] R. Di Lauro, F. Lucarelli, R. Montella. SIaaS - Sensing Instrument as a Service Using Cloud Computing to Turn Physical Instrument into Ubiquitous Service. IEEE 10th International Symposium on Parallel and Distributed Processing with Applications (ISPA). **(2012)**July 10-13,

[12] F. Lelli. Bringing Instruments to a Service-Oriented Interactive Grid. Dipartimento di Informatica, Università Ca’ Foscari di Venezia. Venice,Italy. 2007.

[13] G. C. Fox, R. Guha, D. F. McMullen, A. F. Mustacoglu, M. E. Pierce, A. E. Topcu, and D. J. Wild, Web 2.0 for Grids and e-Science[M]. New York: Springer, 2009.

[14] E. Frizziero, M. Gulmini, F. Lelli, G. Maron, A. Oh, S. Orlando, A. Petrucci, S. Squizzato, and S. Traldi. Instrument Element: a new grid component that enables the control of remote instrumentation. Sixth IEEE International Symposium on Cluster Computing and the Grid. **(2006)**May 16-19,

[15] I. M. Atkinson, D. d. Boulay, C. Chee, K. Chiu, P. Coddington, A. Gerson, T. King, D. F. McMullen, R. Quilici, P. Turner, A. Wendelborn, M. Wyatt, and D. Zhang. "Developing CIMA-based cyberinfrastructure for remote access to scientific instruments and collaborative e-Research." Conferences in Research and Practice in Information Technology Series, Ballarat, VIC, Australia, 2007 pp. 3-10.

[16] M. Okon, D. Kaliszan, M. Lawenda, D. Stokosa, T. Rajtar, N. Meyer, and M. Stroinski. Virtual laboratory as a remote and interactive access to the scientific instrumentation embedded in Grid environment. 2nd IEEE International Conference on e-Science and Grid Computing. **(2006)**December 4-6, Amsterdam, Netherlands

[17] M. Yuriyama, T. Kushida, M. Itakura. A New Model of Accelerating Service Innovation with Sensor-Cloud Infrastructure. Annual SRII Global Conference (SRII). **(2011)**March 29-April 2,

[18] K. Ahmed, M. Gregory. Integrating Wireless Sensor Networks with Cloud Computing. Mobile Ad-hoc and Sensor Networks (MSN), 2011 Seventh International Conference on. **(2011)**16-18 Dec. 2011,

[19] M. S. Aslam, S. Rea, D. Pesch. Service Provisioning for the WSN Cloud. Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. **(2012)**24-29 June 2012,

[20] Y. Byunggu, A. Cuzzocrea, J. Dong, S. Maydebura. On Managing Very Large Sensor-Network Data Using Bigtable. Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on. **(2012)**13-16 May 2012,

[21] Y. Chang Ho, H. Hyuck, J. Hae-Sun, Y. Heon Young, L. Yong-Woo. Intelligent Management of Remote Facilities through a Ubiquitous Cloud Middleware. IEEE International Conference on Cloud Computing. **(2009)**September 21-25,

[22] S. Distefano, G. Merlino, A. Puliafito. Sensing and Actuation as a Service: A New Development for Clouds. 11th IEEE International Symposium on Network Computing and Applications (NCA). **(2012)**August 23-25,

[23] Y. Kang, Z. Wei, H. Song-ling, Discussion on a new concept of measuring instruments-Instrument Cloud, China measurement and test. 38(2), **(2012)**, pp. 1-5.

[24] V. Rajesh, O. Pandithurai, S. Mageshkumar. Wireless sensor node data on cloud. Communication Control and Computing Technologies (ICCCCT), 2010 IEEE International Conference on. **(2010)**7-9 Oct. 2010,

[25] Y. Takabe, K. Matsumoto, M. Yamagiwa, M. Uehara. Proposed Sensor Network for Living Environments Using Cloud Computing. Network-Based Information Systems (NBiS), 2012 15th International Conference on. **(2012)**26-28 Sept. 2012,

[26] G. Zhongwen, L. Chao, F. Yuan, H. Feng. CCSA: A Cloud Computing Service Architecture for Sensor Networks. International Conference on Cloud and Service Computing (CSC). **(2012)**November 22-24,

[27] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, A view of cloud computing, Commun. ACM. 53(4), **(2010)**, pp. 50-58.

[28] T. Hirofuchi, E. Kawai, K. Fujikawa, H. Sunahara, USB/IP: A Transparent Device Sharing Technology over IP Network, IPSJ Digital Courier. 1, **(2005)**, pp. 394-406.

[29] P. Kalagiakos, P. Karampelas. Cloud Computing learning. Application of Information and Communication Technologies (AICT), 2011 5th International Conference on. **(2011)**12-14 Oct. 2011,

[30] Storm. Available: <http://storm.apache.org/>

[31] H. Karimipour, V. Dinavahi, Parallel Domain Decomposition Based Distributed State Estimation for Large-scale Power Systems, IEEE Transactions on Industry Applications. (99), **(2015)**, pp. 1-6.

[32] R. D. Zimmerman, S. Murillo, x, C. E. nchez, R. J. Thomas, MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education, Power Systems, IEEE Transactions on. 26(1), **(2011)**, pp. 12-19.