REAL-TIME CONNECTION BETWEEN IMMERSE 3D VISUALIZATION LABORATORY AND KASKADA PLATFORM

LUKASZ WISZNIEWSKI AND TOMASZ ZIOLKOWSKI

Faculty of Electronics Telecommunication and Computer Science
Technical University of Gdansk
Narutowicza 11/12, 80-233 Gdansk, Poland

(received: 8 June 2015; revised: 7 July 2015;
accepted: 17 July 2015; published online: 1 October 2015)

Abstract: Multimedia stream processing into two cooperative different systems (cluster platform and virtual lab) is considered. The considered selected information about the system is presented and the idea of its communication when executing the distributed application is proposed. A general schema of the communication architecture is given. Tests of data transmission quality are considered and their results are presented.

Keywords: multimedia, cluster computing, visualization laboratory, data transmission

1. Problem statement

During the development of the NIWA Project [1], it was decided to make an interface between the newly built Immerse 3D Visualization Laboratory (I3DVL) [2] and the supercomputer named TRYTON, on which the instance of the KASKADA Platform [3] is running. The main aim of this connection is to provide real-time communication between local IVS computers to use a large high performance clustered computer and features provided by the KASKADA platform, for instance, to render image streams on supercomputer nodes and then project them in the I3DVL. The interface design is presented and its characteristics are evaluated.

1.1. Immerse 3D Visualization Laboratory (I3DVL)

The task of creating a modern virtual reality laboratory, named Immerse 3D Visualization Laboratory, has been started at Faculty of Electronics, Telecommunication and Computer Science of the Technical University of Gdansk. One of the main assumptions of the laboratory is to ensure the highest possible degree of

immersion of feelings (unrestricted freedom of movement and stereoscopic 3D projection) together with the least amount of equipment worn by users (*e.g.* virtual helmets) to provide them with the maximum comfort and impression of natural activity. [4]

The I3DVL is built with a transparent sphere rotating on rollers. A user will be entered into the sphere through a special hatch opened from the outside. In the case of projection on screens surrounding the rotary sphere, high transparency and homogeneity of the sphere is required in order to prevent excessive distortion of the observed image Figure 1. In the I3DVL the rotary transparent sphere with a user inside will be placed in the centre of a cubic cave with edges of about 3.4 meters each. The cave consists of four vertical acrylic flat screen-walls and a horizontal glass screen-floor and a glass screen-ceiling with special coating for highest brightness uniformity. To allow access to the cave, one of these screen-walls will be an automatic sliding door. The 360 degree view will be achieved via the stereoscopic rear projection on the all six flat screen faces forming the cube structure of the cave around the sphere.

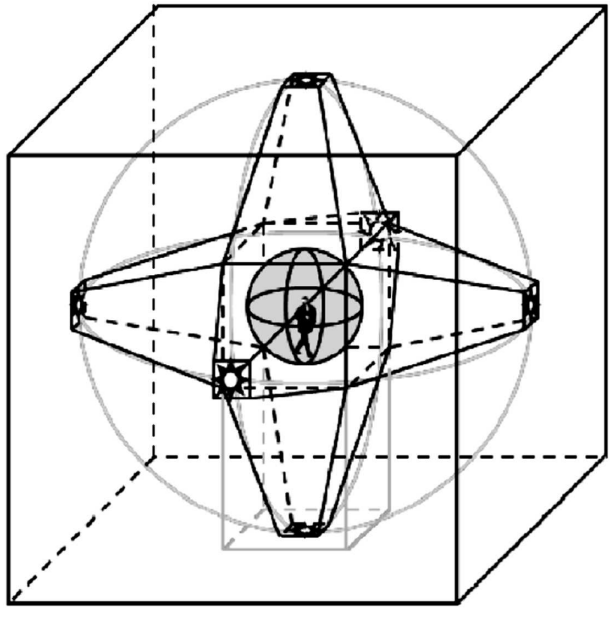

Figure 1. Visualization of the rotary sphere inside the cave

This solution requires projection from six different directions The whole image surround will be displayed by 12 digital projectors – two projectors per screen-wall (floor, ceiling). These three-chip DLP projectors themselves have a resolution of 1920×1200 pixels (WUXGA) and thus a final resolution of single screen-face will be 1920×1920 pixels. Therefore, the square pixel edge length will be 0.067 in (0.17 mm). A luminous flux of the projectors equal to 7000 ANSI

MOST WIEDZY Downloaded from mostwiedzy.pl

lumens implies (taking into account the losses caused by the edge blending) the total luminous flux above 11000 ANSI lumens per screen-wall.

The three-dimensional vision is related to the fact that the eyes of the observer are located at some distance from each other, and therefore each eye catches a slightly different picture. The interpretation of these images takes place in the observer's brain, resulting in the impression of depth. Creating an impression of depth in this manner, requires generation of pairs of images seen from a slightly different perspective, and directing them (*e.g.* by special filter glasses) to the left and right eye of the observer. The stereoscopy with separation in time implemented in the I3DVL uses radio-based active stereo nVidia 3DVision Pro shuttering at 120 Hz. Today's active shutter glasses use polarization, so there is some risk of incorrect operation of glasses caused by possible change of polarization by the rotary sphere material. Therefore, the stereoscopy with spectrum channels separation will be available as an alternative. It will be Barco active Infitec+ using high-quality dynamic color filtering to produce from one projector separate images for the left and the right eye. Visualization is supported by the sound generated by Bose and Apart speakers. The position of the user's head is measured by the ART IR-optical tracking system.

1.2. KASKADA platform and its Framework

The KASKADA Framework [3] is a set of libraries enabling developers to easily implement and deploy data processing algorithms for the KASKADA Platform. Framework-based programs can be executed by the Platform on the Galera's computing nodes. The Framework provides significant functionality for data stream processing, *i.e.*:

- Transmission of KASKADA processing tasks;
- Multimedia stream decoding and encoding;
- Presenting decoded multimedia data samples (along with meta-data) to algorithm implementations, in a form which is easy to analyze and process;
- Converting event streams to their XML representations and sending them to the message broker for further processing.

As both the stream dispatcher and the KASKADA Framework are used for heavy processing, they are written in the C++ language, which allows low processing overhead and low-level access to hardware. Much of the code responsible for the stream processing is shared between the dispatcher and the Framework.

The KASKADA Binary Protocol (KBIN) [5] is a protocol designed for transmitting binary data object streams in the KASKADA Platform. Due to its flexibility, KBIN is used not only to transmit custom data objects, but for all data stream transmission within the Platform. This means that multimedia, event and object streams transmitted between processing tasks are all treated in the same unified way. This fact greatly simplifies the design of data processing algorithms which are deployed in the Platform. KBIN uses the InfiniBand network for data

MOST WIEDZY Downloaded from mostwiedzy.pl

transmission. As KBIN was designed to be easily extensible, it can be used with different existing data transmission interfaces (RDMA, TCP over InfiniBand, MPI over InfiniBand). As mentioned earlier, KBIN transmits data in the form of data objects, *i.e.* objects of various C++ classes. There is a set of predefined classes (*e.g.* Image for video streams, AudioFrame for audio streams and Event for event streams), but developers are free to extend these types or create new ones from scratch. The message broker must mediate between the sender and the receiver during the connection establishment, as some crucial information is unknown to the tasks at launch. It might be, for example, the port number on which the sender listens for connections – it is chosen randomly when the sender initializes its communication channels and must be then sent to the receiver through the message broker. Another reason to use the message broker is that the tasks are initialized in an unspecified order – the message might be sent by the sender before the receiver is ready to accept it. In such a situation, the message would be buffered by the message broker until the receiver finishes its initialization. Once the receiver knows which port to use for the connection, it opens its communication channel and signals the sender (init-connection-ack) through the message broker. It should be noted that this last message cannot be sent to the sender directly through the newly opened KBIN channel, as all KBIN communication channels allow only one-way transmission. The above procedure is valid for connections in complex services – it must be ensured that all communication channels have been properly initialized before the actual data processing (and thus data flow through the channels) can begin. It would be unsuitable, however, to apply the same connection procedure to the stream dispatcher. The dispatcher provides multimedia streams to all services which request it – its lifecycle is independent of the life-cycles of the services. Hence, unlike tasks in complex services, the dispatcher does not know in advance what tasks will require data from him. Therefore, it would be unable to send proper init-connection messages. Instead, the connection parameters for the dispatcher (*e.g.* the port numbers) are stored in the Platform's database. When a service is launched, these parameters are fetched and passed to the new tasks. This way the tasks know what port to use for establishing the connection. By using this procedure, the dispatcher supports dynamic task connection and disconnection. The Platform supports two stream processing modes:

- Synchronized mode In the synchronized mode, all data must be processed in real time. This is often the case when dealing with live stream sources, such as surveillance equipment. When working in the synchronized mode, a certain amount of data is buffered by KBIN channels. In the case of the buffer overflow (when the processing is taking too much time), the excessive data is dropped. Although some data may be lost, this procedure prevents service failures in case of high system load;
- Resynchronized mode In the resynchronized mode, data is processed at the maximum speed allowed by the computing power available to the service. This

MOST WIEDZY Downloaded from mostwiedzy.pl

means that if the system is under heavy load, data is processed at a lower speed. On the other hand, if enough resources are available, the processing speed is increased. This mode is often used when analyzing data from the stream archive, as in many such cases it significantly reduces the analysis time. In the resynchronized mode, KBIN channels do not buffer any data. Instead, data may be sent at different speeds. In this mode, it is guaranteed that no data will be lost during transmission.

In some cases, the C++ object serialization used by the KBIN protocol can cause portability issues. For example, programs built with different compilers or on different hardware platforms may serialize objects in a slightly different way. Therefore, they would be unable to successfully communicate through a KBIN channel in a regular way. For this reason, KBIN supports a portable communication mode. In this mode, KBIN uses serialization routines which are portable between various platforms, at the expense of a possible performance loss.

1.3. Requirements for data transmission

Two scenarios for two different types of data were specified. In the first case (called model) same small data from the I3DVL will be send to the KASKADA Platform. Data will be computed on a supercomputer and the results (also as small data) will be sent back to Laboratory. The received data will be processed for proper projection in the cave. Total bandwidth used here is about dozen of Mbps. The second case (called rendering) is more complex. Computers will send data (also small amounts)from the I3DVL to KASKADA. Data will be used to render multimedia streams which will be sent back to the Laboratory and than presented on screens in the I3DVL. The band which is needed here is:

1 for 2D images:

- 34 Gbps (12 streams @ 1920×1080 px 60 fps);
- 30 Gbps (6 streams @ 1920×1920 px 60 fps);

2 for 3D images:

- 68 Gbps (12 streams @ 1920×1080 px 120 fps);
- 60 Gbps (6 streams @ 1920×1080 px 120 fps).

As we can see, the required bandwidth in the second case is very high. These calculations were made for raw, uncompressed data. In real implementation we can use some compression algorithms, but we want to achieve a real-time (or semi real-time) connection, so that algorithms should not provide too much latency.

2. Interface description

The projected interface is built with two logical layers. The higher and most important level is the functional one, the lower one is the communication level.

2.1. Functional level

This layer must be designed to provide an API which is easy to use for developers, in real-time systems. It must be independent from the communication

MOST WIEDZY Downloaded from mostwiedzy.pl

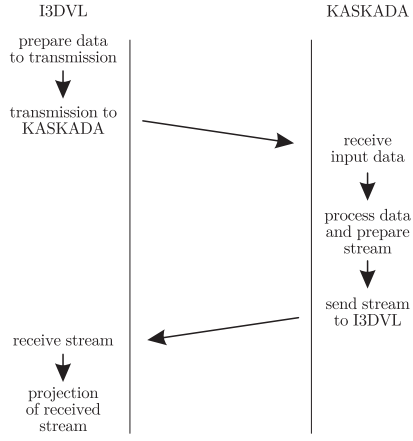

Figure 2. Main idea for exchanging data between I3DVL and KASKADA Platform

level and reduce latency. Figure 2 shows the main idea of data transmission between the I3DVL and the KASKADA Platform.

2.2. Communication level

For the projected connection, we have to use the hardware which is already installed both in TRYTON and I3DVL, and take into account the field obstacles. Academic Computer Centre TASK (where the supercomputer TRYTON is located) and the I3DVL are located in two different buildings, which are 100 meters apart from each other. The connection will be built using a fiber cable.

We can use two transmission technologies and the topology of their connection shown in Figure 3.

2.2.1. Based on Ethernet

The first attempt was to build an Ethernet connection based on the 10Gb technology [6]. The maximum bandwidth between nodes (computers) and switches is 1 Gbps. All nodes in TRYTON are equipped with 1 Gbps interfaces. For the first type of data (model) those speeds are sufficient. For the second case (rendering) the necessary bandwidth between the I3DVL computers and the switch is in the range 2.5–6 Gbps (one computer needs 1/12 of the total bandwidth specified in section 1.2). As we can see this solution is not efficient.

2.2.2. Based on InfiniBand

The second attempt was to try to build a connection based on the InfiniBand Standard [7, 8]. In this case the minimum bandwidth in the proposed network is 40 Gbps which is sufficient for connection between nodes and switches. It is too small for sending raw 3D streams. Nonetheless, it should be sufficient for 2D streams, or 3D streams with a lowered frame rate.

2.2.3. Cooperation between I3DVL and KASKADA Platform

As Figure 4 presets, projected infrastructure based an layered model.

MOST WIEDZY Downloaded from mostwiedzy.pl

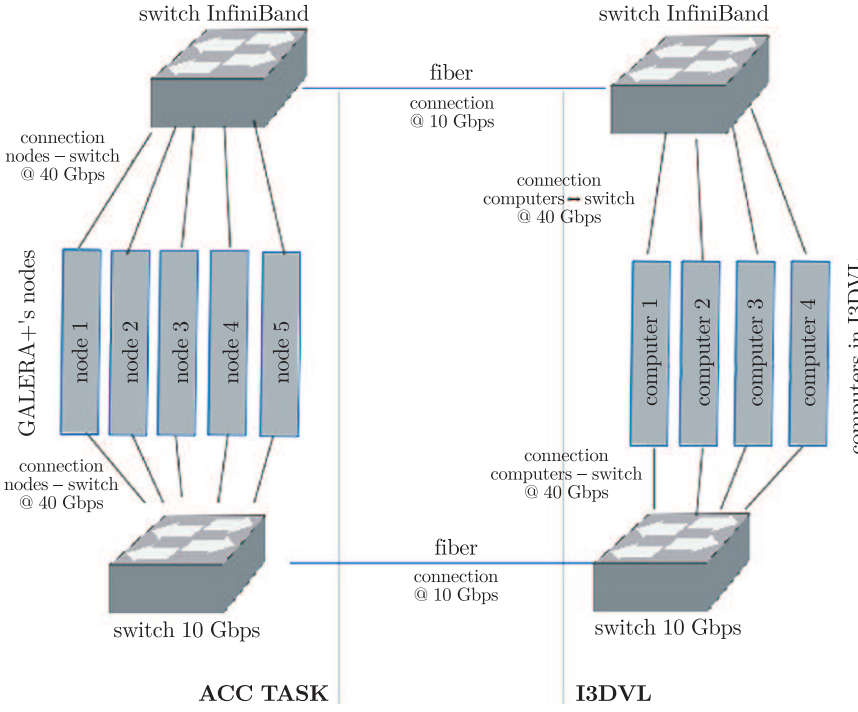

Figure 3. Topology of Ethernet/InfiniBand connection

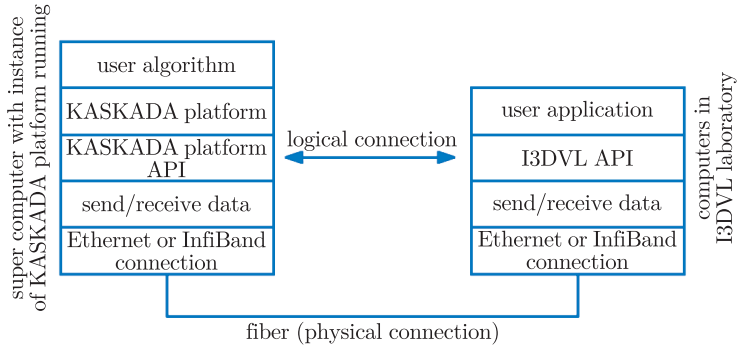

Figure 4. General schema of proposed architecture

On the KASKADA platform there is an algorithm, designed by users, which runs on the KASKADA platform and uses KASKADA API. All the send and receive methods are implemented in API and provide the algorithm developer with a set of methods to manage the sending of data. Below KASKADA API there is an operating system which is responsible for data transmission. On the other end, the architecture in the I3DVL is similar. The user application runs on the operating system, and uses I3DVL API, which is responsible for rendering and projection of images on the cave walls. The operation system is also responsible for managing data stream transmissions.

MOST WIEDZY Downloaded from mostwiedzy.pl

3. Testbed and tests

For further development it was necessary to build an experimental testbed, in which some tests and development could be made. For both technologies (Ethernet and InfiniBand) separate testbeds were created. The main aim was to achieve an architecture as much similar to the real life as possible.

3.1. Testbed Architecture

3.1.1. Testbed for Ethernet

It used one TRYTON node, one normal desktop computer to pretend to be an I3DVL (called "I3DVL") and two 10 Gbit Ethernet switches.

3.1.2. Testbed for InfiniBand

It used four TRYTON nodes. Two of them were used to simulate I3DVL computers and two as KASKADA platforms.

3.2. Testbed experiments

Two tests were performed in the designed testbeds for both transmission technologies. The first test was to measure the latency and the second to verify the bandwidth.

3.2.1. Verify latency

In the Ethernet technology two transmission protocols can be used: TCP and UDP. Both protocols were tested. The test scenario began when the computer which pretended to be an I3DVL started to exchange some data with a TRYTON node. The aim was to measure the latency between the time of sending from "I3DVL" to the TRYTON node and then back to "I3DVL". The tests were performed for UDP and TCP protocols. We used three portions of 10 B, 1 KB and 10 KB data.

Similar tests were performed for the InfiniBand connection. The test scenario began when node 1 started to exchange some data with node 3. We also used three portions of 10 B, 1 KB and 10 KB data.

3.2.2. Verify bandwidth

Theoretically we had up to 1 GBps of a bandwidth.

The TRYTON node was sending data stream increasing their bandwidth from 10 Mbps to 2 Gbps in a 10 Mbps step. The "I3DVL" computer was receiving this data and measured the current data flow. The measured value was the total bandwidth.

Theoretically we had up to 40 Gbps of bandwidth. Nodes 1 and 2 were generating a simple stream to fill the bandwidth. 50 tests were performed. In every test both node 1 and 2 were increasing the transmitted data from 1 Gbps up to 50 Gbps (1 Gbps in the first test, 2 Gbps in the second test, *etc.*). Nodes 3 and 4 were receiving those streams and validated the steam bandwidth.

Maximum bandwidth in Ethernet was measured to 1 Gbps and for InfiniBand to 35 Gbps.

MOST WIEDZY Downloaded from mostwiedzy.pl

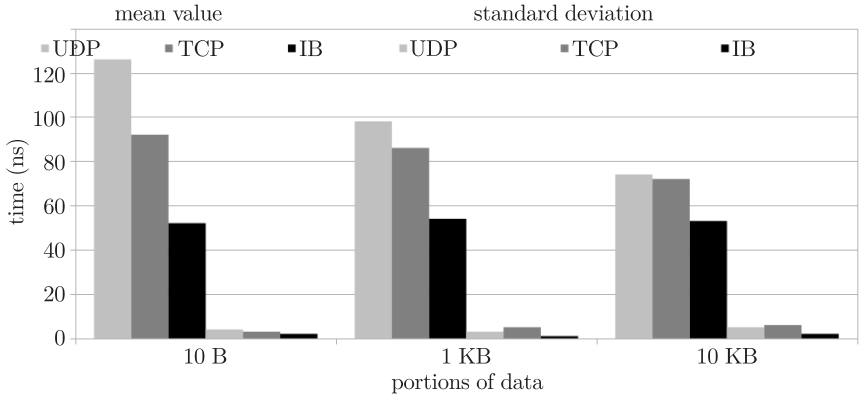

Figure 5. Results for latency in UDP and TCP protocols

As we can see from the above results, both transmission technologies provide very small latency. It seems to be better in InfiniBand, because the standard deviation is smaller than in Ethernet – the results are more similar. The measured maximum bandwidth in InfiniBand is smaller than the theoretical one. The problem is with encoding when sending data. The InfiniBand network uses about 20% of the latency for encoding.

4. Further works

Our aim is to modify the KASKADA Framework to allow programmers to have an easy way to establish connection between the KASKADA Platform and the I3DWL. For the Ethernet base connection there is a need to propose a new protocol to build in the KASKADA Framework. The main idea of communication with this protocol is presented in Figure 4. In a properly designed architecture it should not matter which type of transmission we decide to use. A low-level API which is responsible for managing the transmission must be transparent for high-level user applications. As we can see from the results – the bandwidth in both technologies is too small to send raw streams. There are two possible solutions of this problem: data compression or link aggregation. Data compression will provide some extra latencies in data transmission – firstly it is necessary to compress data before sending, then to decompress them on receiving. Another solution is link aggregation – both Ethernet and InfiniBand technologies support aggregation of several physical links into one logical connection. The maximum bandwidth in this case increases linearly to the number of aggregated links. An optimal number for this connection seems to be 3 links (approx. 105 G bps)

References

[1] *Centrum Doskonałości Naukowej Infrastruktury Wytwarzania Aplikacji (CD NIWA)*, http://niwa.gda.pl/

[2] Lebiedz J and Mazikowski A 2014 *Innovative Solutions for Immersive 3D Visualization Laboratory*, WSCG2014 Conference on Computer Graphics, Visualization and Computer Vision

MOST WIEDZY Downloaded from mostwiedzy.pl

[3] *Supercomputer Platform for Context Analysis of Data Streams in Identification of Specified Objects or Hazardous Events*, http://kaskada.gda.pl
[4] Lebiedz J and Mazikowski A 2013 *Proc. of the 3rd Polish Conference on Computer Games Development WGK* **3** 53 (in polish)
[5] Ziółkowski T 2011 *Management and processing of data streams in the KASKADA platform*
[6] IEEE 802.3$^{\mathrm{TM}}$-2012 *IEEE Standard for Ethernet*
[7] Pentakalos O I 2015 *An Introduction to the InfiniBand Architecture*, O'Reilly
[8] Pfirster G F *An Introduction to the InfiniBand^{TM} Architecture*

MOST WIEDZY Downloaded from mostwiedzy.pl