

Integrating an Attack Tolerant Information Service with Taverna

Erica Y. Yang and Jie Xu

School of Computing
University of Leeds
Leeds LS2 9JT

{ericay, jxu}@comp.leeds.ac.uk

Abstract

Utilising the Grid for conducting scientific research often involves a certain level of security risk, in particular when limited knowledge about the service provider is known before hand. This paper reports our experience of integrating an Attack-Tolerant Information Retrieval (ATIR) service with Taverna, a popular workflow tool among the UK e-science community, to support secure information query in a biology context. This paper presents the system architecture that has been used for the integration and the corresponding implementation details. Performance studies show that the overhead of ATIR server side processing is trivial (<5%) compared with the total processing time of the integrated Taverna. Our experimental results also show that the major processing overhead is caused by the Taverna enactor operations which consume no less than 50% of the total processing time.

1. Introduction

A key stumbling block that often hinders the promise of the Grid is security concerns. When realistic applications of the Grid are discussed, users typically worry about the level of security guarantees they can be certain about the remote service providers. However, as the providers are often discovered on the fly and are used on a session-by-session basis, it is difficult to ensure consistent security assurance. This paper reports the integration of ATIR [YXB03] with Taverna [Tav05], a popular workflow tool among the UK e-science community, to enhance its security so that the privacy of users can be ensured. For the moment, Taverna only comes with a limited support for security. Therefore, to a certain extent, this integration enhances the level of security in the current version of Taverna.

Taverna is particularly popular among biologists who use it to design, enact, and analyse workflows so that large databases, such as genomes datasets, can be efficiently queried and utilised despite the geographical restrictions that may confine biologists. One of the targeted users of Taverna is biologists from pharmaceutical companies who may use it to help drug development and investigation. However, the current version of Taverna

provides limited security features which may be insufficient for such usage. In particular, when a user submits a query which consists of critical information about a drug, service providers can easily infer the intention of the user. However, drug development is a very costly business. The compromise of users' security can cause catastrophic consequence for the user's company. Hence, pharmaceutical companies become cautious about using the (open) Grid to help their investigations despite the fact that drug investigation can require intensive computation resources. Therefore, the dynamic nature of the Grid may compromise the user's confidence of using Taverna for conducting such an activity.

2. An Overview of ATIR

Attack-Tolerant Information Retrieval (ATIR) is a distributed query technique developed and implemented as part of the e-Demand project [eDem05]. ATIR aims to achieve the following goals: i) to protect the privacy of users against accidentally or purposefully information disclosure by information owners; and/or ii) to ensure the correctness of information retrieval against malicious attacks, such as the occurrence of corrupted results.

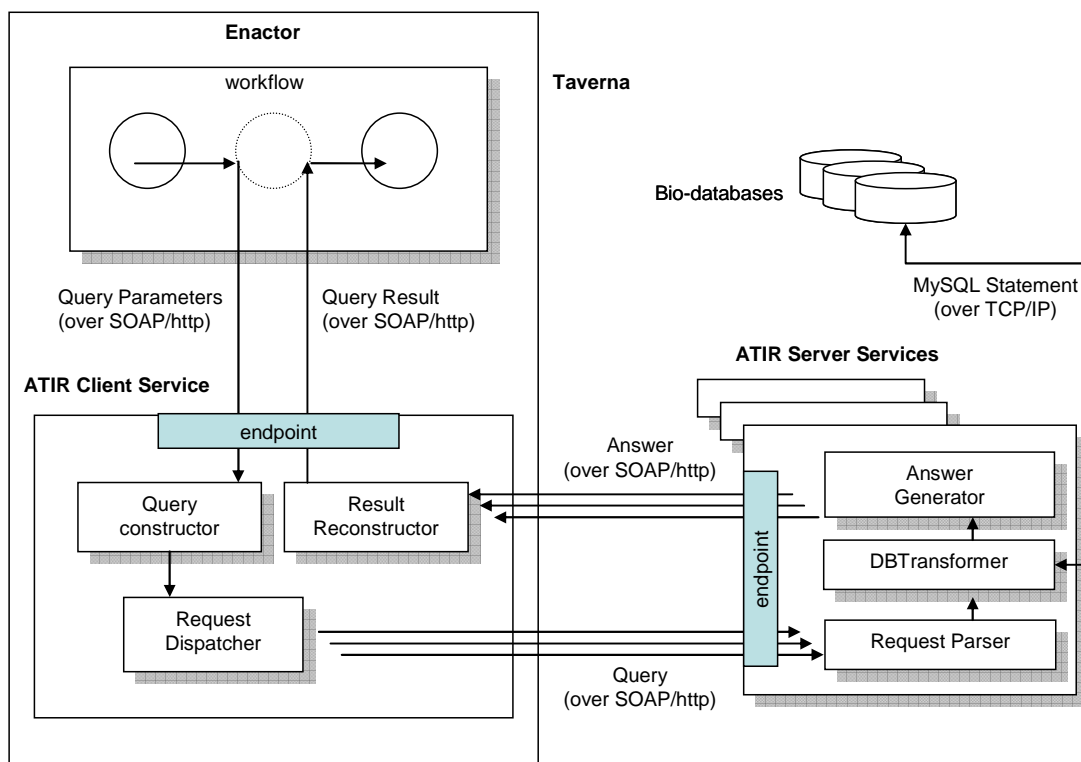


Figure 1 System Architecture of the Integrated Taverna

ATIR exploits replicated services, which are often available in the Grid, to protect a user's privacy and to ensure service availability. This technique is particularly appealing for certain security critical applications in the Grid and Web Services environments, where unknown information resources are often discovered on the fly and are exploited with certain level of security requirements.

ATIR has been implemented based on a client-server architecture where a user utilises an information service through submitting a query to the ATIR client program. Together with a set of randomly generated queries, the client produces a set of query requests which are sent to remote servers to execute respectively. Based upon the replies, the client recovers and identifies the desired result.

3. System Architecture

This section presents the system architecture that has been used to integrate ATIR with Taverna, as illustrated in Figure 1. The figure is divided into three parts: enactor in Taverna, ATIR services, and databases. There are two types of ATIR services in the system: ATIR client service (aka. ATIR client) and ATIR server service (aka. ATIR server). Both services

are implemented as Web services which have public interfaces manifested as WSDL files and can be accessed through their public endpoints via SOAP over HTTP protocol.

The ATIR client is implemented as a processor in Taverna which takes query parameters from the previous component in a workflow and outputs a result to the next component in the workflow. (We shall examine the details of query parameters in Section 4.). The query parameters are wrapped into a SOAP message which is passed to the ATIR client. ATIR client has three components: query constructor, request dispatcher, and result reconstructor. The query constructor is responsible for building query messages based on the parameters from the previous Taverna processor. The request dispatcher initiates multiple threads, each of which sends the query request to remote ATIR servers accordingly. Similarly, the dispatcher also wraps the queries into SOAP messages and transmits them over HTTP protocol to remote servers.

Each ATIR server also contains three components: request parser, DBTransformer, and answer generator. Based on a query message, the parser generates a SQL query statement that is passed into the DBTransformer as an input. The main task of DBTransformer is

to convert the character-based result set obtained from the query statement into a number-based result set. In this paper, we define the size of a result set as the number of records in the result set. This new result set is used as an input to the answer generator to compute an answer for sending back to the ATIR client. The result reconstructor on the client then reconstructs the answers into a result which is passed back to the workflow as an input to the next component.

In Taverna, processors are primary components in a workflow and each of them is atomic in the sense that they are independent from each other. This property means that different processors can be easily integrated within a workflow despite the concrete implementation techniques inside each processor. As shown in the figure, we adopt the same principle. The details of ATIR implementation are hidden from the other processors in the workflow. Ideally, other processors should make no awareness of the use of the ATIR processor. In reality, when choosing an ATIR processor as part of a workflow, the user should decide the level of privacy protection required and the number of replicated services involved. In Taverna, the enactor is an abstracted component that coordinates data and execution flows of the processors in a workflow.

4. System Implementation

This section describes the implementation details of ATIR client and ATIR server. We will explain how to integrate an ATIR processor as part of a Taverna workflow.

4.1 A Simple Workflow of an ATIR Processor

Figure 2 shows a simple workflow of an ATIR processor. It takes six query parameters as workflow inputs, they are: number of curious servers that the system can tolerate (t), minimum number of servers (k_{pi}), the size of a result set (s), valid range of data ($data_range$), intended data item ($keyword$), and intended record columns ($field$). The size of a result set is a parameter that is chosen by the user (i.e. the previous workflow component) to determine the level of privacy protection that is required. For example, when s is set to be 100, it means that the server side computation will be performed over 100 records. Although the user only intends for one of the records, the server will not know exactly which one the user is after

(apart from the fact that the intended one must be one of the hundred records).

For t , k_{pi} , $data_range$, the typical settings are: 1, 2, and 255, respectively. As explained earlier in Section 2, the answers are numbers that are resulted from the calculation of the answer generator. The calculation is carried out over a certain range of data that is used by both client and servers. The calculation carried out by the result reconstructor is also over this data range. The calculation range is divided into two categories: valid or invalid range. A calculated number can only be in one of such range. Whenever a calculated number is beyond the $data_range$ specified previously, this number is deemed to be invalid.

The keyword and field are used to define the type of the intended information that a user wants to obtain from databases. For example, the keyword could be a gene name and the field can be the columns in the databases.

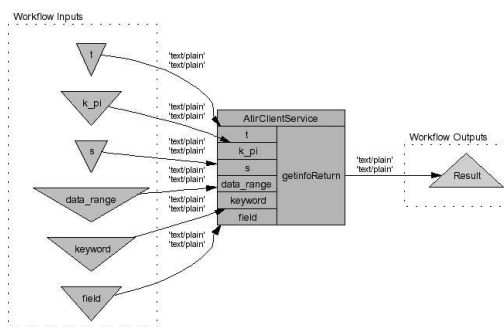


Figure 2 A Simple Workflow of the ATIR Processor

5. Performance Evaluation

5.1 Experimental Environment

All the experiments described in this section are conducted in a close LAN environment. We use a set of identical DELL machines with the following specifications: Dell Precision 650 workstation with Dual Intel Xeon 3.06GHz hyper-threaded, 1 GB RAM, 36GB Fujitsu MAS3367NP SCSI hard drive, and 3COM 3c905C NIC. They are interconnected through a Cisco switch which has the following specification: Cisco Catalyst 2924 (model: WS-C2924C-XL-A). All machines run Redhat Linux release 9 (Shrike) and the kernel version is 2.4.26. The kernel is compiled with smp support. The specifications of the major software packages are described as follows. The version of MySQL is 11.18 with distribution 3.23.58 for redhat-linux-gnu (i386).

ATIR client and servers are deployed in the environment as Web Services on different machines.

5.2 Experimental Objectives and Analysis Model

In order to evaluate the performance of the integrated Taverna, we first establish three objectives for our experimental studies:

- Find out the overall time consumption of using the integrated Tavera;
- Evaluate the extra overhead involved in various ATIR operations; and
- Identify the performance bottleneck of the integrated Taverna.

We use the typical settings of t , k_{pi} , and $data_range$, as described in Section 4, and incrementally increase the value of s from 10 to 100 with 10 more records added each time.

To satisfy the above experimental objectives, we derive two analysis models for our experiments. Since ATIR client and servers are all deployed as Web services, we write a java program, called *atircd_main*, which directly invokes the ATIR client service exactly as Taverna does, but without going through the ATIR Taverna enactor. Practically, that means *atircd_main* does not have the WSDL/XML parsing overhead of Taverna enactor. With this program, we can compare two types of invocations: direct invocation (through *atircd_main*) and indirect invocation via Tavera enactor. Therefore, we can have a better understanding of the exact overheads involved in using the integrated Taverna. The analysis model one is for measuring the direct invocation cost and is described as follows:

$$T_{am} = T_{wp} + T_{ac} + T_{as}$$

T_{am} is the time taken for *atircd_main* to invoke an ATIR client service. T_{wp} is the time consumed by SOAP message generating and parsing. T_{ac} is the time taken by the ATIR client service, that is, the total processing time of three components: query constructor, request dispatcher, and result reconstructor. T_{as} is the time taken by the ATIR server side processing, that is, the total processing time of three components: request parser, DBTransformer, and answer generator.

The analysis model two is for measuring the indirect invocation cost via the integrated Taverna and is as follows:

$$T_t = T_{tp} + T_{wp} + T_{ac} + T_{as}$$

T_t is the time taken for the integrated Taverna to invoke an ATIR client service. T_{tp} is the time consumed by Taverna enactor for WSDL/XML parsing. T_{wp} , T_{ac} , T_{as} are the same as those in the model one.

In our models, we ignore the SOAP message transmission time. This is based on two grounds: 1. all the experiments are conducted in a LAN environment; 2. all the messages transmitted over the network are less than 1k. To simplify the analysis, we therefore eliminate the cost of message transmission in the above models.

5.3 Processing Times of the Integrated Taverna

Figure 3 shows the processing time of the integrated Taverna. The X-axis is the number of records involved in the server side processing. The top line shows the changes of the total processing time along with the increasing of record numbers. As shown in the figure, the total processing time (TPT) fluctuates significantly from time to time. This is mainly due to the system level garbage collection performed by Java Virtual Machines (JVMs). The invocation of the integrated Taverna involves four independent JVMs connected via the network. Since garbage collection is managed by individual JVMs, the aggregated effect caused by garbage collection is evident. In contrast, since the performance of ATIR client (the middle line) and ATIR server (the bottom line) are measured on a single machine. As shown in the figure, the measurements become relatively stable. This doesn't mean that there is little effect of Java garbage collection on client and server machines, but only indicates that the impact of garbage collection is much smaller compared with that to overall performance cost of the integrated Taverna.

Figure 3 also shows that the proportions of ATIR client and server processing time are relatively small in the total processing time of Taverna. Consistently across all measurements, ATIR client is less than 40% of the total processing time. The time taken by all three components of ATIR server is very small. It takes less than 13 milliseconds (less than 5% of the total processing time) to complete all the processing on the server side on all the experiments we performed. Although we would expect this number to arise as the number of records is further increased, we expect the proportion of the server side processing in the overall processing time to be relatively small.

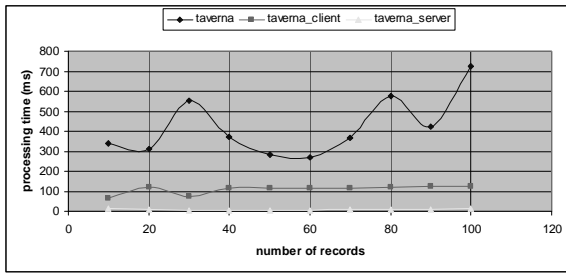


Figure 3 Processing Time of Integrated Taverna

Also from Figure 3, it is also clear that the major cost of using ATIR service is on the client side rather than the server side.

Figure 4 shows various processing time of using our own Java program `atircd_main` to invoke ATIR services. Different from Figure 3, this figure indicates that the total processing time is mainly dictated by the time taken by the client side processing. This is because the direct invocation does not involve any extra WSDL/XML generating and parsing as those needed by the indirect invocation using the Taverna enactor. Similar to the server processing time in Figure 3, the server side processing time is also a small proportion of the total processing time in the direct invocation.

5.4 Overhead of Using Taverna Enactor

In order to find out the overhead of using Taverna Enactor to invoke ATIR service (i.e. T_{tp}), we can make use of the analysis models presented in 5.2 because we have:

$$T_{tp} = T_t - T_{am}$$

Table 1 shows the overhead of using Taverna Enactor to invoke an ATIR service. Compared with using `atircd_main` alone, more than 50% of the total processing time of the integrated Taverna is consumed because of the use of the enactor for WSDL/XML parsing and generating.

Table 1 Overhead of Using Taverna Enactor

s	T _{am}	T _t	T _{tp} /T _t
10	157.42	338.9	54%
20	92.8	310.9	70%
30	126.5	551.6	77%
40	128.2	371.8	66%
50	127	282.8	55%
60	130.9	270.3	52%
70	126	367.3	66%
80	130.3	578.3	77%
90	153.5	421.9	64%
100	173.1	726.5	76%

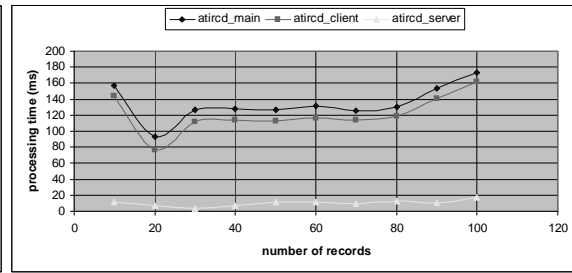


Figure 4 Processing Time of `atircd_main`

6. Conclusions

This paper describes an application of applying the ATIR technology developed in the e-Demand project to a realistic e-Science workflow tool - Taverna. We demonstrate its applicability through a thorough description of the system architecture that has been used for the integration and conduct significant performance studies of the implemented system. The results show that the performance overheads of conducting ATIR operations are not significant in the overall total processing time of the integrated Taverna. We also found that the cost of using Taverna enactor to invoke ATIR services is at least 50% higher than directly invoke them. This suggests that the convenience and flexibility brought by an integrated workflow tool can impose a significant amount of performance overhead. However, this is well expected as there is always a tradeoff between usability and performance.

References

- [eDem05] The e-Demand Project, <http://www.comp.leeds.ac.uk/edemand/>.
- [Tav05] The Taverna Project, <http://taverna.sourceforge.net/>.
- [YXB03] E. Y. Yang, J. Xu and K. H. Bennett, "Sharing with Limited Trust: An Attack-Tolerant Service in Durham e-Demand Project", (regular paper), U.K. e-Science 2nd All-Hands Meeting, Simon J. Cox Eds., Nottingham Conference Center, U.K., Sept. 2nd - 4th, 2003, ISBN 1-904425-11-9.