Comparison of JSON and XML Data Interchange Formats: A Case Study

Nurzhan Nurseitov, Michael Paulson, Randall Reynolds, Clemente Izurieta

Department of Computer Science
Montana State University – Bozeman
Bozeman, Montana, 59715, USA
{nurseitov@cs.montana.edu, mpaulson@cs.montana.edu,
rreynolds@cs.montana.edu, clemente.izurieta@cs.montana.edu}

Abstract

This paper compares two data interchange formats currently used by industry applications; XML and JSON. The choice of an adequate data interchange format can have significant consequences on data transmission rates and performance. We describe the language specifications and their respective setting of use. A case study is then conducted to compare the resource utilization and the relative performance of applications that use the interchange formats. We find that JSON is significantly faster than XML and we further record other resource-related metrics in our results.

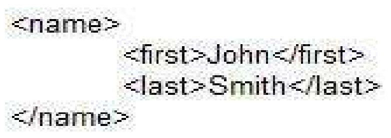
1. INTRODUCTION

Data interchange formats evolved from being mark-up and display-oriented to further support the encoding of meta-data that describes the structural attributes of the information. The requirements to support data interchange of Java applications led to the development of standard data interchange formats [2]. JSON and XML are two data interchange formats with unique purposes. Sections two and three provide background for JSON and XML. Section four describes the case study and methodology used to compare speed and resource utilizations. Section five describes results and section six identifies the threats to the validity of this study. We conclude in section seven and provide directions for possible refinements to this study.

2. XML

The Extensible Markup Language (XML) [3] is a subset of the Standard Generalized Markup Language (SGML) [8] and evolved as a result of the complexity of SGML. XML is considered the 'holy grail' of computing due to its universal data representation format [1]. The intent of an XML document is self evident and embedded in its structure. The fundamental design considerations of XML include simplicity and human readability. Amongst the design goals of XML, the W3C specifies that *"XML shall be straightforwardly usable over the Internet"* and *"XML documents should be human-legible and reasonably clear."* [3]

The primary uses for XML are Remote Procedure Calls (RPC) [4] and object serialization for transfer of data between applications. XML is a language used for creating user-defined markups to documents and encoding schemes. XML does not have predefined tag sets and each valid tag is defined by either a user or through another automated scheme. Vast numbers of tutorials and user forums provide wide support for XML and have helped create a broad user base. XML is a user-defined hierarchical data format. An example of an object encoded in XML is provided in figure 1.



```
<name>
        <first>John</first>
        <last>Smith</last>
</name>
```

Figure 1: A hierarchical structure describing the encoding of a name

3. JSON

JSON [6] is designed to be a data exchange language which is human readable and easy for computers to parse and use. JSON is directly supported inside JavaScript [7] and is best suited for JavaScript applications; thus providing significant performance gains over XML, which requires extra libraries to retrieve data from Document Object Model (DOM) [12] objects. JSON is estimated to parse up to one hundred times faster than XML [6] in modern browsers, but despite its claims of noteworthy performance, arguments against JSON include lack of namespace support, lack of input validation and extensibility drawbacks. Crockford [6] addresses such arguments by claiming that *"every object is a namespace. Its set of keys is independent of all other objects, even exclusive of nesting. Also, JSON uses context to avoid ambiguity, just as programming languages do,"* that validation of inputs is the responsibility of individual domain applications, and that the lack of extensibility claims is addressed by the flexibility of JSON constructs.

JSON's syntax is human readable. Figure 2 describes an example where JSON is used to encode a *firstname* and a *lastname*.

```
{
        "firstname" : "John",
        "lastname" : "Smith"
}
```

Figure 2: A simple JSON construct describing the encoding of a name

4. METHODOLOGY

This case study measures transmission times and resource utilizations. The null hypothesis states that there is no difference in transmission times and resource utilization between JSON and XML. The operational environment for this case study consists of a client/server program. The client is setup in isolation and sends JSON and XML objects to the server in order to measure performance and resource utilization. We find significant evidence to support rejecting the null hypothesis.

4.1. Client/Server Program

Our test cases use a simple network client program to transmit XML-encoded and JSON-encoded Java objects to a server. The client and server initiate TCP/IP based connections where the server listens on a port and the client connects on that port. Similar coding techniques are used by both the client and server. To simulate realistic servers and potentially run stress tests, the server is multi-threaded. The server decodes the JSON or XML text upon receipt and then discards the text.

4.2. Environment

The client program sends JSON and XML encoded data in an isolated workbench environment. The hardware consists of two workstations interconnected by a switch. Software firewall services are disabled. Since data interchange does not involve reading and writing to secondary storage, disk read/write capability is not important to these workstations. The workstations have a CentOS 5.2 [14] minimal installation with additional software packages to record various system measures. The workstations are connected to an isolated local area network. The switch supports gigabit connections and the workstations have 10/100 megabit network interface cards. Cat5e cables run between the workstations' network interface cards and the switch, and the full topology is arranged within close proximity (less than 10 meters). According to the network monitoring tool, IPTraf [5], our isolated network does not show frequent network broadcast traffic.

4.3. Measurements

We choose to measure the following metrics: number of objects sent, total time to send the number of objects, average time per object transmission, user CPU utilization, system CPU utilization, and memory utilization. The total time per trial tells us how long it takes for the server to receive every object from the client. The average time per object describes how long it takes (on average) for the server to receive one object from the client. The user CPU utilization is the percentage of time spent performing user processes and the system CPU utilization is the percentage of time spent performing system processes. According to RedHat [9], high user CPU percentages tend to be favorable while high system percentages tend to point towards problems that will require further investigation [13]. Memory utilization measures the percentage of available and free memory on the system. Our metrics are recorded to files using client/server software developed in-house and System Activity Reporter (SAR), a metric recording utility [11]. The client/server program measures transmission time per-object-transmission. SAR measures resource utilizations per-time-duration.

Timing measurements are computed as follows. The client connects and sends a start command to the server. When the server receives the start command, it starts a timer and sends a ready message. The client receives the ready message and begins the transmission of objects. When the client is finished transmitting its objects, it sends an end signal to the server and the server turns off its timer and its log metrics. Metrics are recorded into a file with a timestamp to indicate when the trial completes.

4.4. Case Study Design

Test cases are designed and implemented to compare transmission times and resource utilizations of JSON and XML. The first scenario consists of running a single time-consuming transmission of a large quantity of objects in order to achieve accurate average measurements. The second scenario consists of running a series of test cases with increasingly higher number of objects. Its purpose is to determine if JSON or XML differ statistically as the number of encoded objects sent to the server increases. The number of objects transmitted to the server is treated as an independent variable. By increasing the number of objects sent to the server at equally spaced discrete intervals, we add variance to the distributions of the measurements for the mean-comparison t-test. Additionally, we compare the impact of transmitting a high number of objects with the impact of transmitting a low number of objects by observing what happens to measurements at varying degrees of granularity.

The first scenario consists of a client sending one million objects to a server using both JSON encoding and XML encoding. The second scenario consists of a client sending smaller quantities of objects to a server in five separate intervals. The client sends 20,000, 40,000, 60,000, 80,000, and 100,000 encoded objects to the server. We refer to these transmission intervals as Trial 1, Trial 2, Trial 3, Trial 4, and Trial 5, respectively.

5. RESULTS

Results illustrate the differences between JSON and XML encoding under varying transmission scenarios. This section presents the metrics obtained for the average measurements, compares the metrics of transmitting high versus low number of encoded objects, and determines whether JSON and XML are statistically different for each of our measurements. We present both scenarios' measurements and discuss their implications.

5.1. Scenario 1

Scenario 1 is a time-consuming transmission of a large quantity of objects. Large numbers of objects are used in order to achieve accurate average measurements. The client sends one million encoded objects to the server for both JSON and XML. We measure timing and resource utilizations. Tables 1 and 2 list the measurements and respective values obtained from this trial:

Table 1: Scenario 1 JSON vs. XML Timing

	JSON	XML
Number Of Objects	1000000	1000000
Total Time (ms)	78257.9	4546694.78
Average Time (ms)	0.08	4.55

Table 2: Scenario 1 JSON vs. XML CPU/Mem

	Average % User CPU Utilization	Average % System CPU Utilization	Average % Memory Utilization
JSON	86.13	13.08	27.37
XML	54.59	45.41	29.69

5.2. Scenario 2

Scenario 2 is comprised of a series of smaller trials that determine whether JSON and XML are statistically different according to each of our measures. The mean-comparison t-test is used. We send 20,000, 40,000, 60,000, 80,000, and 100,000 encoded objects to the server and collect metrics for each case. Tables 3, 4 and 5 display the metrics obtained from these trials:

Table 3: Scenario 2 JSON Vs XML Timing

	JSON	XML
Trial 1 Number Of Objects	20000	20000
Trial 1 Total Time (ms)	2213.15	61333.68
Trial 1 Average Time (ms)	0.11	3.07
Trial 2 Number Of Objects	40000	40000
Trial 2 Total Time (ms)	3127.99	123854.59
Trial 2 Average Time (ms)	0.08	3.10
Trial 3 Number Of Objects	60000	60000
Trial 3 Total Time (ms)	4552.38	185936.27
Trial 3 Average Time (ms)	0.08	3.10
Trial 4 Number Of Objects	80000	80000
Trial 4 Total Time (ms)	6006.72	247639.81
Trial 4 Average Time (ms)	0.08	3.10
Trial 5 Number Of Objects	100000	100000
Trial 5 Total Time (ms)	7497.36	310017.47
Trial 5 Average Time (ms)	0.07	3.10

Table 4: Scenario 2 JSON CPU/Mem

Trial	Average % User CPU Utilization	Average % System CPU Utilization	Average % Memory Utilization
1	29.07	14.80	67.97
2	83.84	15.84	68.07
3	88.01	11.99	68.06
4	88.65	11.36	68.06
5	88.70	11.30	68.06

Table 5: Scenario 2 XML CPU/Mem

Trial	Average % User CPU Utilization	Average % System CPU Utilization	Average % Memory Utilization
1	65.80	32.36	68.08
2	67.43	32.57	68.08
3	66.69	33.31	68.08
4	67.24	32.76	68.11
5	66.64	36	68.79

Figure 3 illustrates JSON's average CPU and memory utilizations per trial. Figure 4 illustrates XML's average CPU and memory utilizations per trial. Figure 5 illustrates the differences between JSON's resource utilizations and XML's resource utilizations by plotting Figure 3 and Figure 4 on the same graph. Figures 3-5 indicate that XML appears to use less user CPU utilization than JSON. JSON and XML encoded transmissions use nearly the same amount of memory on the server.

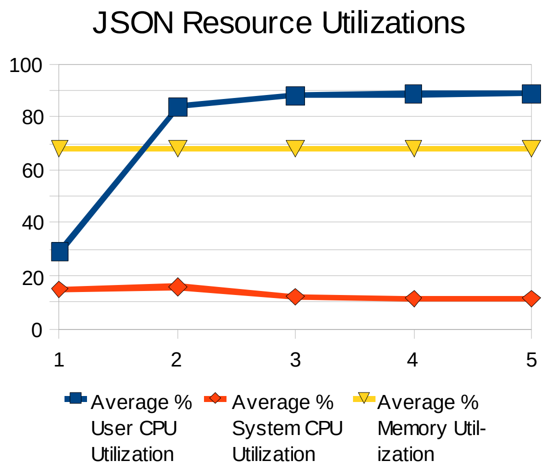


Figure 3: Scenario 2 JSON Resource Utilizations

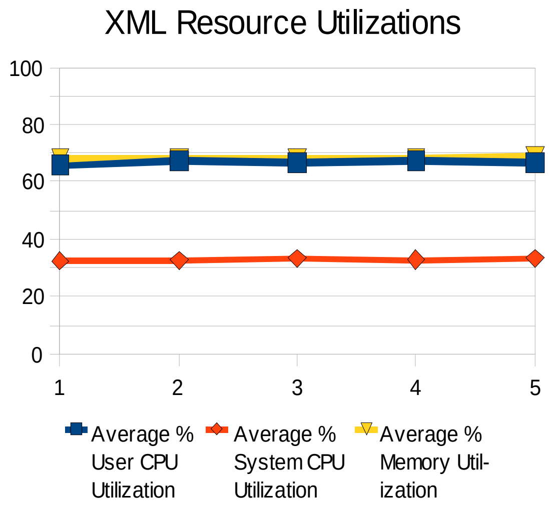


Figure 4: Scenario 2 XML Resource Utilizations

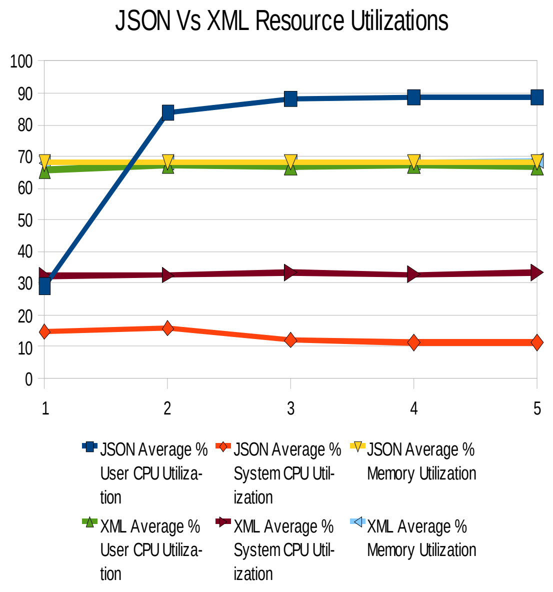


Figure 5: Scenario 2 JSON Vs XML Resource Utilizations

5.3. Discussion

To analyze our results we make high-level qualitative observations and further analyze each measure's significance using statistical tests. This section explains the observed differences using a traditional t-test.

High-level qualitative observations between JSON and XML are observed from both test scenarios. Scenario 1 illustrates accurate average measurements because of the high number of encoded object transmissions. Scenario 2 provides fine grained observations of the impacts of fewer transmissions for each measurement. Table 6 lists the differences between JSON and XML based on the observations and the results of each scenario.

The average values of measurements from scenario 1 indicate that sending data in JSON encoding is in general faster than using XML encoding. The average time and total time measures provide an indication that JSON's speed outperforms XML's speed. In addition, JSON uses more user CPU resources than XML in scenario 1. Memory is dependent on the state of the systems before a scenario or trial execution; however, usage is similar between JSON and XML. According to our observations and the metrics obtained from both scenarios, the transmission times of XML are lower when fewer objects are transmitted and the transmission times of JSON are the same when fewer objects are transmitted. The transmission of a lower quantity of JSON-encoded objects does not appear to impact the user CPU utilization measure when compared to the transmission of a higher quantity of objects.

Table 6: High-Level Results and Observations

Scenario/Measure	JSON	XML
Scenario 1 Total Time	78.26 seconds	75.77 minutes
Scenario 1 Average Time Per Object	0.08 ms	4.55 ms
Scenario 1 Average User CPU Utilization	86%	55%
Scenario 1 Average System CPU Utilization	13%	45%
Scenario 1 Average Memory Utilization	27%	29%
Scenario 2 Total Time for 100,000 Objects	7.5 seconds	310 seconds
Scenario 2 Average Time Per Object	0.08 ms	3.1 ms
Scenario 2 Average User CPU Utilization	83-88%	65-67%
Scenario 2 Average System CPU Utilization	11-14%	32-33%
Scenario 2 Average Memory Utilization	68%	68%

The t-test is a way to compare the means of two distributions and determine whether they are statistically different. We run a two-sided unpaired t-test on the results of scenario 2 to compare JSON and XML with respect to each measure. The level of significance is set at $\alpha = 0.05$ [10]. The distribution of each measure is the set comprising all five observations that come from each of the five trials in scenario 2. We make the null hypothesis assumption that JSON and XML have the same means for each measure distribution. Then, we use the t-test to calculate the probability that would provide evidence to support an alternate hypothesis. Table 7 lists the sample distributions used in the t-tests to compare each measure. The distribution values come from tables 3, 4 and 5. Table 8 lists the probabilities (p-values) that we would have come to our results under the null hypothesis assumption.

Table 7: JSON and XML sample populations used in the t-test

Measure	JSON Distribution	XML Distribution
Total Time (ms)	{2213.1, 3127.99, 4552.38, 6006.72, 7497.36}	{61333.68, 123854.59, 185936.27, 247639.81, 310017.47}
Average Time Per Object (ms)	{0.11, 0.08, 0.08, 0.08, 0.07}	{3.07, 3.10, 3.10, 3.10, 3.10}
Average % User CPU	{29.07, 83.84, 88.01, 88.65, 88.70}	{65.80, 67.43, 66.69, 67.24, 66.64}
Average % System CPU	{14.80, 15.84, 11.99, 11.36, 11.30}	{32.36, 32,57, 33,31, 32,76, 33.36}
Average % Memory	{67.97, 68.07, 68.06, 68.06, 68.06}	{68.08, 68.08, 68.08, 68.11, 68.79}

Table 8: JSON and XML t-test p-values with $\alpha = 0.05$

Measure	p-value
Total Time (ms)	**0.0033**
Average Time Per Object (ms)	**≤ 0.0001**
Average % User CPU	0.47
Average % System CPU	**0.0038**
Average % Memory	0.23

T-test results show that JSON and XML have statistically different total time per trial, average time per trial, and average system CPU utilization per trial.

6. THREATS TO VALIDITY

Case studies are subject to validity threats, specifically, internal validity, external validity, construct validity and content validity. We use measures that adequately represent notions of performance and resource utilization as described in section 4.3, thus providing meaningful metrics of construct validity. The setup of our test scenarios where the client and server programs are isolated from broadcast traffic in a network provide for additional confidence that our metrics are not confounded by additional variables. Various measures are used to understand differences between JSON and XML; thus increasing the content validity of the study.

Internal validity refers to the relationship that exists between independent and dependent variables. The measures described in section 4.3 represent the dependent variables of our study, and the only independent variables are the test case programs running under scenarios 1 or 2. The homogeneity and isolation of test cases running under JSON or XML increases the internal validity of the study.

External validity refers to the ability to generalize results from this case study. Clearly this is not possible as additional test cases would be necessary to account for different operating systems, content of data, package sizes transmitted over the network, etc. This case study serves as a single data point to demonstrate performance and resource utilization differences between JSON and XML for the given specific test cases.

7. CONCLUSION

This case study compared the differences between two current data interchange formats. Results indicate that JSON is faster and uses fewer resources than its XML counterpart; thus providing significant evidence to refute the null hypothesis.

JSON and XML provide unique strengths, but the importance of performance and resource utilization must be understood when making decisions between data interchange formats. This case study has provided a clear benchmark that can be used to compare these formats when selecting a transport mechanism. We intend to continue and improve our investigations as follows: 1) eliminate the potential network bottleneck in our test scenarios by using gigabit network cards. Gigabit network cards give us the ability to perform stress-based cases to see which data interchange format handles multiple connections more effectively. Gigabit network cards also give us the ability to obtain case metrics while the server is under heavy load, and 2) perform a survey to compare the learning times of JSON and XML to see which format has a steeper learning curve.

8. REFERENCES

[1] T. Anderson, 2004. http://www.itwriting.com/xmlintro.php

[2] J. Bosak, *"Xml, java, and the future of the web,"* World Wide Web Journal, 2(4):219-227, 1997.

[3] Extensible markup language (xml) 1.0 (fourth edition). W3C, 2006. http://www.w3.org/TR/2006/REC-xml-20060816

[4] U. Hilger, *"Article: Client/server with java and xml-rpc,"* 2005. http://articles.lightdev.com/csjxml/csjxml_article.pdf.

[5] G. P. Java. Iptraf - an ip network monitor. IPTraf, 2001. http://iptraf.seul.org

[6] JSON. json.org. http://www.json.org

[7] S. Klarr, *"Javascript: What is json?,"* 2007. http://www.scottklarr.com/topic/18/javascript-what-isjson

[8] J. F. E. v. d. V. D. A. J. D. A. W. L. M. David Hunter, Jeff Rafter, *"Beginning xml,"* 4th edition, pp. 6-8, 2007.

[9] Redhat.com — the world's open source leader. redhat.com, 2009. http://www.redhat.com

[10] Student's t-tests. physics.csbsju.edu, 2009. http://www.physics.csbsju.edu/stats/t-test.html

[11] System activity reporter (SAR). Softpanorama, 2009. http://www.softpanorama.org/Admin/Monitoring/sar.html

[12] W3C Document Object Model. W3C, 2005. http://www.w3.org/DOM

[13] *"What to monitor? Red Hat Linux 9: Red Hat Linux System Administration Primer,"* 2003. http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/admin-primer/s1-resource-what-to-monitor.html

[14] www.centos.org - the community enterprise operating system, CentOS, 2005. http://www.centos.org