A Comparative Performance Evaluation of Multi Processor Multi Core Server Processor Architectures on Enterprise Middleware Performance

W.M. Roshan Weerasuriya and D.N. Ranasinghe
University of Colombo School of Computing, Colombo, Sri Lanka
E-mail: {wmr, dnr}@ucsc.lk Tel: +94-112-158978

***Abstract*-In this paper we describe the performance evaluation and comparison of server based "Enterprise Middleware" frameworks on multi-processor multi-core server processor architectures. We experimented a 'single processor quad core Intel Xeon' server processor and a 'dual processor dual core multiprocessor AMD Opteron'. Also we discuss the expected enterprise middleware framework execution performance of the two micro-architectures by analyzing the statistics obtained from the respective microprocessor technical data sheets. Our experiment results show that the "single processor Xeon" gives a better throughput than the "multiprocessor Opteron". With this study we found that the 'single processor quad core Intel Xeon' server processor outperforms the 'dual core dual processor AMD Opteron' server processor in executing server based enterprise middleware applications. Hence we see that out of the two evaluated server processor architectures the single processor multi-core architecture is giving better performance than the multi-processor multi-core architecture, in executing enterprise middleware frameworks.**

I. Introduction

In this study we compare and analyzed the performance of executing enterprise middleware applications by two server microprocessors namely the Dual-Core AMD Opteron 2220 SE microprocessor [2,7] and the Quad Core Intel Xeon E5506 [4,5]. The AMD server contained two Dual-Core AMD Opteron 2220 SE microprocessors (i.e. a total of four cores), which makes it a suitable competitor with the four core Intel Xeon E5506.

We developed our benchmark testing programs using the Java language [13], Java Beans, and using Enterprise Middleware frameworks: JavaEE (JSP, Session Beans, Entities) [14], Struts[15], Spring[16], Hibernate[17]. Apache commons DBCP[18] was used to handle database connection pooling. The test programs were compiled using Sun Java 1.6.0_20 on Ubuntu 10.04 kernel (Linux 2.6.32), 64 bit operating system. The application server used is JBossAS 6.0.0.Final [19]. The Java Runtime is OpenJDK Runtime Environment (build 1.6.0_20-b20).

For load generation and load testing we used Apache JMeter [12] Load Testing tool. We stressed the server processors running the enterprise middleware, by generating different levels of HTTP loads.

Our approach for evaluating the server processor performance is to analyze the statistics obtained from Apache JMeter. Also we analyze the enterprise middleware execution performance with respect to the microprocessor hardware statistics which we have obtained from the microprocessor technical data sheets.

Our findings revealed that the "single processor quad core Intel Xeon" microprocessor performs better than the "dual core multiprocessor (dual processor) AMD Opteron" microprocessor, with respect to executing enterprise middleware frameworks. Hence we see that out of the two evaluated server processor architectures the single processor multi-core architecture is giving better performance than the multi-processor multi-core architecture, in executing enterprise middleware frameworks.

II. Microprocessor Architectural Specifications

In this section we summarize the micro architectural level details of the two microprocessors, which we gathered by going through the microprocessor data sheets and technical documents of the two microprocessors. This summarization helps the researchers to know as of what micro architectural level parameters are important to look in to, when it comes to analyzing the performance of microprocessors.

TABLE I
MICROPROCESSOR DETAILS [1,2,3,4,5,6,7,8,9]

	Processor	Code Name	Release Date, Price at introduction	Multi Core	64bit (x86 64)
Dual-Core AMD Opteron	2220 SE	Santa Rosa	Aug 15, 2006, $1165	Dual (2)	y
Intel Xeon Quad Core	E5506	Nehalem-EP, Gainestown	March 30, 2009, $266	Quad (4)	y

Table II
Microprocessor Details [1,2,3,4,5,6,7,8,9]

	Out-of-order execution, speculative execution	Super-scalar execution	Hyper Threaded (SMT)	Hyper Transport	SIMD
Dual-Core AMD Opteron	y	y	n	y	y
Intel Xeon Quad Core	y	y	n	n	y

Table III
Microprocessor Details [1,2,3,4,5,6,7,8]

	speed(GHz)	L1 cache (KB)	L2 cache	L3 cache	Bus Speed
Dual-Core AMD Opteron	2.8	(64 data + 64instru = 128) x per core	1MB x per core	no	1000 (MT/s)
Intel Xeon Quad Core	2.13	(32 data + 32instru = 64) x per core	256 KB x per core	4 MB (Intel Smart Cache, Shared)	2400 MHz QPI

Table IV
Micro-architecture Parameters [3, 9, 10, 11]

	AMD Opteron 2220 SE	Intel Xeon E5506
	3 way super-scalar processor	
No of pipeline stages	12 for integer 17 for floating-point	14
Reorder buffer	72 entry	128 entry
No of decoders	3 (“fastpath” decoders)	4
can decode up to how many instructions in one cycle?	3 instructions (Fastpath decoder) 1 instruction (Microcode decoder)	4 μops
can issue \ dispatch up to how many μops per cycle	11 micro-ops (The schedulers and the load/store unit can dispatch). 3 micro-ops to the instruction control unit	6 μops
can retire up to how many instructions per cycle?	3	4 μops (or up to 5 with macro-fusion)
No of arithmetic logical units (ALU)	3 integer, 3 floating-point	3, 3
Up to how many floating-point operations per cycle	3 floating-point units, which each can retire 1 instruction	8
No of issue ports available to dispatching SIMD instructions for execution		3
No of integer general purpose registers (GPRs)	16	
No of streaming SIMD extension (SSE) registers	16	

Table V
Integrated Memory Controller [3, 8]

	AMD Opteron 2220 SE	Intel Xeon E5506
	Dual channel 128-bit wide	6-channel
	333 MHz DDR memory	800 MHz DDR memory
Peak memory bandwidth	5.3 Gbytes/s	up to 25.6 GB/sec

Table VI
Memory access: Load and Store Operation Enhancements [3, 10 pg 88]

	AMD Opteron 2220 SE	Intel Xeon E5506
Address support \ size	40 bits physical 48 bits virtual	40 bits physical 48 bits virtual
Peak issue rate operation per cycle	Two 64-bit loads or stores	one 128-bit load one 128-bit store
Load-to-use latency	3 cycles	
Load/ store queue	44-entry	Deeper buffers for load and store operations: 48 load buffers 32 store buffers 10 fill buffers

The above summarized architectural specifications from Table I to VI shows that configuration wise each microprocessor has its own strengths compared to the other.

III. Implementation

We implemented six benchmark testing programs using different enterprise middleware solutions as following:

- a JSP, Java Beans test (Program 1)
- a JSP, JavaEE Session Beans test (Program 2)
- a JSP, JavaEE Session Beans, JPA Entity classes test (Program 3)
- a Struts MVC test (Program 4)
- a Struts MVC, Hibernate test (Program 5)
- a Spring MVC, Hibernate test (Program 6)

TABLE VII
CODE SEGMENTS FROM THE TESTING PROGRAM 2 WRITTEN

```
From the .jsp
<%
        InitialContext ic = new InitialContext();
        SaveSessionBeanLocal saveLocal = (SaveSessionBeanLocal)
ic.lookup("TestProject2Ear/SaveSessionBean/local");

        boolean bStatus = saveLocal.save();
        …
%>
From the Session Bean
@Stateless
public class SaveSessionBean implements SaveSessionBeanLocal {
        private Employee employee = new Employee();
        private EmployeeDAO employeeDAO = …;
…
public boolean save() {
        employee.setName("taro");
        employee.setSalary(20000.0);
        bStatus = employeeDAO.saveEmployee(employee);
        if (bStatus) {
                return true;
        } else {
        setErrorMessage(employeeDAO.getErrorMessage());
        …
From the Data Access Object
public class EmployeeDAOImpl implements EmployeeDAO {
        public EmployeeDAOImpl() {
                Class.forName("com.mysql.jdbc.Driver");
                con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/middleWareR
esearch" ,"root", "");
…
public boolean saveEmployee(Employee employee) {
        stmt.executeUpdate("insert into employee2 (name, salary)
values (…);
                …
From the Domain class
public class Employee implements Serializable {
        public Employee(Integer empNo, String name, double salary)
        {
                this.empNo = empNo;
                …
```

TABLE VIII
CODE SEGMENTS FROM THE TESTING PROGRAM 3 WRITTEN

```
From the Session Bean
@Stateless
public class SaveSessionBean implements … {
        @PersistenceContext
        EntityManager em;
…
From the Data Access Object
public boolean saveEmployee(Employee employee, EntityManager em) {
                em.persist(employee);
        …
From the Domain class
import javax.persistence.*;
@Entity
@Table(name="employee3")
@SequenceGenerator(name = "emp_sequence", …)
public class Employee implements Serializable {
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY,
generator = "emp_sequence")
        public Integer getEmpNo() {
                return empNo;
From the Data Source configuration
<?xml version="1.0" encoding="UTF-8"?>
<datasources>
  <local-tx-datasource>
    <jndi-name>MySQL_DS</jndi-name>
    <connection-
url>jdbc:mysql://localhost:3306/middleWareResearch</connection-url>
 …
```

TABLE IX
CODE SEGMENTS FROM THE TESTING PROGRAM 5 WRITTEN

```
From the JSP
<%@taglib uri="/struts-tags" prefix="s"%>
<s:action name="addEmployee" executeResult="true"></s:action>
From the struts.xml file
<struts>
<package name="default" extends="hibernate-default">
        <action name="addEmployee" method="add"
class="pk1.web.EmployeeAction">
                <result name="success">save.jsp?
status=afterSave</result>
…
From the hibernate.cfg.xml file
<hibernate-configuration>
<session-factory>
        <property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</proper
ty>
        <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/middleWare
Research</property>
        <property
name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
        <mapping class="pk1.domain.Employee" />
        …
From the Action class
import com.opensymphony.xwork2.ActionSupport;
public class EmployeeAction extends ActionSupport {
        public String add(){
                employeeDAO.saveEmployee(employee);
        …
From the Data Access Object
import org.hibernate.Session;
import
com.googlecode.s2hibernate.struts2.plugin.annotations.SessionTarget;
public class EmployeeDAOImpl implements EmployeeDAO {
        @SessionTarget
        Session session;
        @TransactionTarget
        Transaction transaction;
        @Override
        public void saveEmployee(Employee employee) {
                session.save(employee);
                 ...
```

TABLE X
CODE SEGMENTS FROM THE TESTING PROGRAM 6 WRITTEN

```
From the dispatcher-servlet.xml
<?xml version="1.0" encoding="UTF-8"?>
<beans …>
        <bean id="viewResolver"
class="org.springframework.web.servlet.view. …>
        <bean id="myDataSource"
class="org.apache.commons.dbcp.BasicDataSource" …>
                …
        </bean>
        <bean id="mySessionFactory"
class="org.springframework.orm.hibernate3.annotation.AnnotationSession
FactoryBean">
```

```
                        <property name="dataSource"
ref="myDataSource" />
                        <property name="annotatedClasses">
                                    <list>
                                    <value>pk1.domain.Employee</value>
                                    </list>
                        </property>
                        …
            </bean>
            <bean id="myEmployeeDAO"
class="pk1.dao.EmployeeDAOImpl">
                        <property name="sessionFactory"
ref="mySessionFactory"/>
            </bean>
            <bean name="/*.htm" class="pk1.web.EmployeeController" >
                        <property name="employeeDAO"
ref="myEmployeeDAO" />
            </bean>
</beans>
```

We deployed our packaged Web Archive (.war) and Enterprise Archive (.ear) modules to the JBoss Application Server.

MYSQL was used as the database server.

Apache commons DBCP was used as the database connection pooling library to handle the concurrent requests to the database.

We used Apache JMeter as our load testing tool. In order assess the middleware performance we generated the following different types loads to stress the middleware frameworks running on JBoss Application Server:

– 500 concurrent requests within 1 second ramp up time period
– 1000 concurrent requests within 1 second ramp up time period
– 1500 concurrent requests within 1 second ramp up time period
– 1500 concurrent requests within 5 seconds ramp up time period

For the above different loads we accessed the following server statistics:

– average time taken to handle a request
– standard deviation of request handling
– error % of request handling
– throughput of request handling

IV. Results, Evaluation, Analysis and Discussion

Table XI
Performance Statistics Obtained for Program 1

Processor, No of concurrent requests, Ramp up time	Average	Std Dev	Error %	Throughput / Sec
Opteron, 500req, in1Sec	71	129	0	239.46
Xeon, 500req, in1Sec	17	40	0	317.66
Opteron,1000req, in1Sec	5	6	0	604.59
Xeon, 1000req, in1Sec	2	8	0	739.09
Opteron,1500req, in1Sec	4036	3878	6.8	136.68
Xeon, 1500req, in1Sec	0	0.65	0	1076.81
Opteron,1500req, in5Sec	44	128	0	270.12
Xeon, 1500req, in5Sec	0	0.58	0	285.33

Table XII
Performance Statistics Obtained for Program 2

Processor, No of concurrent requests, Ramp up time	Average	Std Dev	Error %	Throughput / Sec
Opteron, 500req, in1Sec	27	46	0	341.29
Xeon, 500req, in1Sec	405	346	0	341.99
Opteron,1000req, in1Sec	57	59	0	505.05
Xeon, 1000req, in1Sec	10	24	0	740.74
Opteron,1500req, in1Sec	212	205.72	0	321.4
Xeon, 1500req, in1Sec	1541	1452	0	338.6
Opteron,1500req, in5Sec	82	191.49	0	262.51
Xeon, 1500req, in5Sec	2144	2821	0	135.18

Table XIII
Performance Statistics Obtained for Program 3

Processor, No of concurrent requests, Ramp up time	Average	Std Dev	Error %	Throughput / Sec
Opteron, 500req, in1Sec	70	45	0	361.53
Xeon, 500req, in1Sec	20	35	0	340.13
Opteron,1000req, in1Sec	2956	1480	0	93.53
Xeon, 1000req, in1Sec	1127	1362	0	211.32
Opteron,1500req, in1Sec	3203	2899	27	128.27
Xeon, 1500req, in1Sec	2930	3108	35	152.06
Opteron,1500req, in5Sec	2962	3524	0	104.34
Xeon, 1500req, in5Sec	2717	3968	0	108.57

TABLE XIV
PERFORMANCE STATISTICS OBTAINED FOR PROGRAM 4

Processor, No of concurrent requests, Ramp up time	Average	Std Dev	Error %	Throughput / Sec
Opteron, 500req, in1Sec	82	97	0	245.94
Xeon, 500req, in1Sec	21	37	0	374.53
Opteron,1000req, in1Sec	1265	1441	0	201.12
Xeon, 1000req, in1Sec	1	2	0	779.42
Opteron,1500req, in1Sec	2006	1990	0	121.77
Xeon, 1500req, in1Sec	2070	1997	0	142.45
Opteron,1500req, in5Sec	4357	4711	6	66.72
Xeon, 1500req, in5Sec	1929	3051	0	126.89

TABLE XV
PERFORMANCE STATISTICS OBTAINED FOR PROGRAM 5

Processor, No of concurrent requests, Ramp up time	Average	Std Dev	Error %	Throughput / Sec
Opteron, 500req, in1Sec	321	211	0	297.08
Xeon, 500req, in1Sec	73	58	0	298.68
Opteron,1000req, in1Sec	9274	7727	25	44.22
Xeon, 1000req, in1Sec	3662	3310	0	95.48
Opteron,1500req, in1Sec	933	1043	0	220.07
Xeon, 1500req, in1Sec	8066	8942	56	67.86
Opteron,1500req, in5Sec	13754	11482	78	43.79
Xeon, 1500req, in5Sec	8146	7793	47	60.98

TABLE XVI
PERFORMANCE STATISTICS OBTAINED FOR PROGRAM 6

Processor, No of concurrent requests, Ramp up time	Average	Std Dev	Error %	Throughput / Sec
Opteron, 500req, in1Sec	46	60	0	321.33
Xeon, 500req, in1Sec	3	6	0	418.06
Opteron,1000req, in1Sec	1638	1459	0	206.99
Xeon, 1000req, in1Sec	1	2	0	711.23
Opteron,1500req, in1Sec	264	367	0	348.27
Xeon, 1500req, in1Sec	1786	1421	27	364.25
Opteron,1500req, in5Sec	9383	8357	40	59.01
Xeon, 1500req, in5Sec	4500	5599	6	64.02

Table V summarizes few parameters of the Integrated Memory Controllers of the two server microprocessors. By analyzing the Integrated Memory Controllers of the two microprocessors we could see the Intel Xeon has the better unit out of the two microprocessors, which will make the Xeon to perform well in executing enterprise middleware frameworks.

Analyzing Table VI we could see that when looking at Memory access: Load and Store Operation Enhancements, the Xeon parameters are ahead than the Opteron enhancements, which will make the Xeon to perform well.

Analyzing Table III we could see that marginally the Opteron has the better cache parameters out of the two processors. Also the Opteron has the higher clock speed out of the two processors. This should make the enterprise middleware to perform well on dual processor Opteron server. But looking at our final results we see that the enterprise middleware has performed well on the single processor Xeon server.

Our results from Table XI to XVI shows that in executing enterprise middleware frameworks, in overall the single processor quad core Xeon processor gives a better throughput than the dual processor dual core Opteron processor.

When the processing loads of the enterprise middleware frameworks were increased (i.e. the number of concurrent requests were increased) the performance of the Xeon degraded by a visible margin and the performance of Opteron came very close to the performance of the Xeon. I.e the enterprise middleware running on Opteron server processor starts to give similar performance as with the enterprise middleware running on Xeon server processor, for higher workloads. But still for higher workloads the throughput of enterprise middleware running on Xeon is better than the Opteron.

V. CONCLUSIONS

In our previous work done [20,21] we carried out a series of performance evaluation experiments using the server based microprocessors: 'dual processor dual core AMD Opteron 2220 SE' and 'single processor quad core Intel Xeon E5506', in order to analyze their performance in executing different software application types such as: thread based, matrix multiplication, processing intensive, system call intensive applications, file reading and writing, socket based, message passing middle ware based and memory intensive applications. Our previous work findings did show that the Opteron perform better in the application categories: memory intensive applications and processing intensive applications. Xeon performed better in the application categories: system call applications, file reading and writing, socket based applications and in thread based applications. In executing server based enterprise middleware frameworks the single processor Xeon clearly outperformed the dual processor Opteron.

This research paper summarizes our research findings with respect to executing server based enterprise middlware frameworks by multi-processor multi-core server processors. The research findings of this paper shows us that the single processor quad core Xeon gives a better throughput than the dual processor dual core Opteron, in executing server based enterprise middlware applications, hence we see that out of the the two evaluated server processor architectures the single processor multi-core architecture is better than the multi-processor multi-core architecture, for executing enterprise middleware frameworks.

Acknowledgment

We thank AMD cooperation for their generosity in providing us the Opteron server resources for this research. This work is supported by the Large Scale Enterprise Applications research group and High Performance Computing research group.

References

[1] AMD processors for servers and workstations, http://products.amd.com/en-ca/OpteronCPUDetail.aspx?id=319&f1=&f2=&f3=&f4=&f5=&f6=&f7=&f8=&f9=&f10=&

[2] Builder's guide for AMD OpteronTM processor-based servers and workstations

[3] Chetana N. Keltcher, Kevin J. McGrath, Ardsher Ahmed, Pat Conway, "The AMD Opteron processor for multiprocessor servers", IEEE Micro archive, Volume 23 Issue 2, March 2003

[4] Intel Xeon processor E5506, http://ark.intel.com/Product.aspx?id=37096

[5] Intel Xeon processor 5500 series datasheet, volume 1

[6] Billy Brennan, Christopher Ruiz, Kay Sackey, "Intel Xeon Nehalem architecture", University of Virginia. www.cs.virginia.edu/~skadron/cs433_s09_processors/nehalem.ppt

[7] AMD second generation Opteron 2220 SE, http://www.cpu-world.com/CPUs/K8/AMD-Second%20Generation%20Opteron%202220%20SE%20-%20OSY2220GAA6CQ%20%28OSY2220CQWOF%29.html

[8] Intel Xeon E5506, http://www.cpu-world.com/CPUs/Xeon/Intel-Xeon%20E5506%20-%20AT80602000798AA%20%28BX80602E5506%29.html

[9] Henry Cook, Kum Sackey, Andrew Weatherton, "The AMD Opteron", www.cs.virginia.edu/~skadron/cs451/opteron/opteron.ppt

[10] Intel 64 and IA-32 architectures optimization reference manual, Order Number: 248966-023a, January 2011

[11] Understanding the detailed architecture of AMD's 64 bit core, http://www.chip-architect.com/news/2003_09_21_Detailed_Architecture_of_AMDs_64bit_Core.html

[12] Apache JMeter, http://jakarta.apache.org/jmeter/

[13] Java Language, http://www.java.com/en/

[14] Java Enterprise Edition, http://download.oracle.com/javaee/

[15] Struts Framework, http://struts.apache.org/2.x/index.html

[16] Spring Framework, http://www.springsource.org/

[17] Hibernate Framework, http://www.hibernate.org/

[18] Apache Commons DBCP, http://commons.apache.org/dbcp/

[19] JBoss Application Server, http://www.jboss.org/

[20] W.M.R. Weerasuriya and D.N. Ranasinghe, "Older Opteron Outperforms the Newer Xeon: A Memory Intensive Application Study of Server Based Microprocessors", 21st International Conference on Systems Engineering (ICSEng 2011), Las Vegas, NV, USA.

[21] W.M.R. Weerasuriya and D.N. Ranasinghe, "Performance Analysis of System Call Intensive Software Application Execution on Server Processor Architectures: Opteron and Xeon", 2nd International Conference on Emerging Trends in Engineering and Technology (IETET-2011), Kurukshetra (Haryana) India. in press.