Multi-Channel Clustered Web Application Servers

Masters Thesis Proposal

Progress

American University in Cairo

Proposed by Karim Sobh (kmsobh@aucegypt.edu)
Supervised by Dr. Ahmed Sameh (sameh@aucegypt.edu)

November, 2008
Outline

• Quick Introduction and Previous Presentation Overview
• Work Done Before Proposal
• Work Done After Proposal
• Cluster Management Subsystem
• Deployment Work flow
• What is remaining?
• Questions
Quick Introduction

Objective

• Targets gain in **Performance** through **Clustering**, **Distribution**, and **Parallelization** of web applications and enabling such features in a framework from which web application servers can inherit
• Investigate **bottle necks** in current Web Application Servers and Web applications conventions that if changed will lead to better performance
• Design such architectural changes with **Transparency** in mind.
• Extend already existing development technologies and build on to of them. (e.g. in-line web scripting techniques)
• Borrow techniques for the **HPC** field that can be applicable to the domain of web application and utilize them as much as possible.
• The architectural changes proposed should act as a **better underlying environment** for current web environment’s **non-performance** features such as **business logic isolation**, **interoperability**, and **expandability**, **extendibility**, **fault tolerance**, and ease of **deployment**.

November, 2008
Quick Introduction

Research Questions

What is the effect, on performance, of changing the communication layer of web environments from the normal stateful TCP to the stateless UDP communication protocol, and how will this change help the emergence of new mechanisms and features that will add to performance and to already established characteristics of web environments such as expandability, extendibility, fault tolerance, high availability, ... etc. ?
Quick Introduction

Deliverable

Performance Deliverables
   Measured experimental results for different architectures comparison. (TCP With dispatchers versus multi-channel over UDP)

Feature Deliverables
   Additional features that are gained due to the architectural change of the network communication layer, and may add to performance in some situations.
Work Done Before Proposal

Some work has been done before the proposal presentation to verify the idea on the following different fronts:

- Design
- Prototyping
- Preliminary Experiments
Design

The main and important use cases, class diagrams, sequence diagrams, and deployment diagrams are designed and ready for implementation.

UML2 Specification is used as a Design Modeling Language.

Visual-Paradigm case tool is used as a design case tool.
A prototype for a C++ container is implemented with the following features:
• Service Loader and Factory
• UDP multi-threaded Communication Layer
• Service and communication channels Garbage Collector
• XML Configuration Reader

A Prototype for HPA is implemented with the following features:
• Multi-threaded UDP Communication Layer
• Multi-threaded TCP Communication Layer
• Server Page parser
• XML Configuration Reader
Work Done Before Proposal

Preliminary Experiments- Single Channel

• Pure Throughput test through single channel file serving.
• Experiment was applied on an HPA/C++ Container in comparison with the Apache/PHP
• Network Speed is 100 Mb/Sec
• The experiment is performed over several runs with different workloads.
Work Done Before Proposal (5/8)
Preliminary Experiments- Single Channel Results

Average Time (Sec)

Max Client Service Time (Sec)

Average Client Bandwidth (MB/sec)

Minimum Client Bandwidth (MB/sec)

Fluctuation Time (Sec)

Fluctuation Bandwidth (MB/sec)

Total Bandwidth (MB/sec)

Total Experiment Duration (Sec)

Fluctuation Bandwidth Gain for Client (MB/sec)

Average Bandwidth % Gain Per Client

November, 2008 Multi-Channel Clustered Web Application Servers
Work Done Before Proposal

Experiments- Multi-Channel

• 2 Channel Container Service running on a cluster of 2 nodes and one HPA.
• Apache Running on a cluster of 2 nodes and one dispatcher.
• Network Speed is 100 Mb/Sec
• The experiment is performed over several runs with different workloads and different processing delays.

November, 2008
Work Done Before Proposal

Results – Multi-Channel

% in Bandwidth Gain Per Client

2 Clients

% in Bandwidth Gain Per Client

4 Clients

% in Bandwidth Gain Per Client

6 Clients

% in Bandwidth Gain Per Client

8 Clients

% in Bandwidth Gain Per Client

10 Clients

% in Bandwidth Gain Per Client

12 Clients

% in Bandwidth Gain Per Client

14 Clients

November, 2008

Multi-Channel Clustered Web Application Servers
Let $N$ be the normal service delivery time on current web Application Servers Environments:

$$N = \text{Service Execution Time} + \text{Service Data Transfer Time}$$

Let $M$ be the service delivery time on Multi-channel Environment

$$M = \text{Max Channel (Execution + Data Transfer )} + \text{HPA Overhead}$$

Let $C$ be the number of channels

$$M = \left( \frac{N}{C} \right) \times W + \text{HPA Overhead}$$

Where $W$ is the weight of the maximum channel ranging from 1 to $C$. For service with an equally distributable channels $W = 1$.

Therefore, if we define $P$ as the gain in performance, then:

$$P = (N - M) + K + D$$

Where $K$ is the difference in data transfer between TCP and UDP
And $D$ is the difference in data transfer resulting from caching static content within dynamic scripts.
Work Done After Proposal

- C++ Implementation Finalization
- Java Container Full Implementation
- C++ Cluster Management Subsystem.
- Java Cluster Management Subsystem.
- HPA Cluster Management Subsystem Client.

November, 2008
Cluster Management

Services

- Cluster Communication Layer
  - Native Multi-cast
  - Emulated Multi-cast
- Reservation Service
- Discovery Service
- HPA Discovery Client
- Replicated Shared Memory
- Performance Data Replication
Cluster Management

Container Node Types

• Compute Node
  ➢ Cluster Communication Layer
  ➢ Channel Reservation Service
  ➢ Replicated Shared Memory Service
  ➢ Performance Data Replication Service

• Management Node
  ➢ Discovery Service
  ➢ Deployment Administrative Tools
  ➢ In addition to Compute Node Services
Deployment Manager

- Server Side Script Parser
- Development Language Identification
- Server Side Services Code Generation
- Server Side Services Code Compilation
- Server Side Services Node Distribution
- Server Page Skeleton Map Construction
- Server Page Skeleton Map Node Distribution
Deployment Manager
Class Hierarchy

The Ancestor class is the abstract class provided by the container and all the services that will be deployed by the container should inherit/extend this class.

The Ancestor has a lot of methods, but the most important one is the main method which is the method called by the dispatcher to start the execution of the service, so it is the service body.

The serialize and deserialize methods are there to be overridden by classes of services that target the utilization of the service migration feature, and they are implemented by the developer of the service to serialize all or some of the data members of the service before migration.

The serialize and deserialize methods are called by the inter-cluster interface subsystem during the migration of services.

The recipient node of the migrated service should have the service implementation compiled and service instances loaded by the service factory.
Deployment Manager
Deployment Process

Server Side Script
Provided By The Developer

Skelaton
Map

Scanner

Source Code

Consolidator

Java Compiler

C++ Compiler

Cluster Manager

C++ Container Node

C++ Container Node

JAVA Container Node

C++ Container Node

JAVA Container Node

November, 2008
Multi-Channel Clustered Web Application Servers
Deployment Manager

Example

```c
#include "FILE1.h"

FILE1::FILE1 () : Ancestor()
{
}

void FILE1::main (HTTPRequest * httpRequest , HTTPReply * httpreply)
{
    if (output == NULL)
    {
        printf("No output handler\n");
        return;
    }

    FILE * f = fopen("/root/install1_large.log", "rt");
    char buffer [CHUNK_SIZE];
    memset (buffer, 0, CHUNK_SIZE);
    int readSize = fread (buffer, 1, CHUNK_SIZE, f);
    for (; readSize > 0;)
    {
        if (readSize < CHUNK_SIZE)
        {
            output->echo (buffer, readSize, true);
        }
        else
        {
            output->echo (buffer, readSize); // echo data
            memset (buffer, 0, CHUNK_SIZE);
            readSize = fread (buffer, 1, CHUNK_SIZE, f);
        }
    }
    close (f);
}

FILE1::~FILE1 ()
{
}

Ancestor * create_object ()
{
    FILE1 * x = new FILE1 ();
    return x;
}
```
<table border="1">
  <tr>
    <td>
      <pre>&lt;?container (THREAD1)\nGET /FILE1?filename=http_filename1 HTTP/1.1
Host: 10.0.0.2:9010
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
?&gt;</pre>
    </td>
    <td>
      <pre>&lt;?container (THREAD2)\nGET /FILE2?filename=http_filename2 HTTP/1.1
Host: 10.0.0.3:9010
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
?&gt;</pre>
    </td>
  </tr>
</table>
This is test file1 content This is test file1 content This is test file1 content This is test file1 content This is test file1 content This is test file1 content This is test file1 content This is test file1 content This is test file1 content This is test file1 content This is test file1 content This is test file1 content This is test file1 content This is test file1 content This is test file1 content This is test file1 content This is test file1 content This is test file1 content This is test file1 content

This is test file2 content This is test file2 content This is test file2 content This is test file2 content This is test file2 content This is test file2 content This is test file2 content This is test file2 content This is test file2 content This is test file2 content This is test file2 content This is test file2 content This is test file2 content This is test file2 content This is test file2 content This is test file2 content This is test file2 content This is test file2 content This is test file2 content This is test file2 content

November, 2008

Multi-Channel Clustered Web Application Servers
What is Remaining

- Finish the Deployment Manager (C++/JAVA)
- HPA Skeleton Caching
- Design Experiments and Benchmarks For:
  - Performance Experiments
  - Fault Tolerance Experiments
  - Hybrid Scripting
- Apply Experiments
- Final Experiment Report
- Final Defense
Questions