Table of Contents

Organization Overview

Tata Steel Limited

Established in 1907 [1] by **Jamshedji Nusserwanji Tata**, Tata Steel is **Asia's first** and **India's largest integrated private sector steel company**. It is one of the few select steel companies that is **EVA+ (Economic Value Added)**. Over the years, Tata Steel has emerged as a thriving, nimble, steel enterprise, due to its ability to transform itself rapidly to meet the challenges of a highly competitive global economy and commitment to become a supplier of choice by delighting its customers with services and products. Constant modernization and introduction of state-of-the-art technology at Tata Steel has enabled it to stay ahead in the industry and successfully meet the expectations of all sections of stakeholders. Tata Steel's four-phase Modernization Programme in the steel works has enabled it to acquire the most modern steel making facilities in the world. Its captive raw material resources and the state-of-the-art **5 MTPA (million tones per annum) plant at Jamshedpur**, in Jharkhand State, India give it a competitive edge. Determined to be a major global steel player, Tata Steel has recently included in its fold **NatSteel, Asia (2 MTPA) and Millennium Steel (now Tata Steel Thailand) (1.7 MTPA)** creating a manufacturing network in eight markets in South East Asia and Pacific rim countries. Soon the Jamshedpur plant will expand its capacity from 5 MTPA to 7 MTPA by 2008. The Company plans to enhance its capacity, manifold through organic growth and investments. The Company's wire manufacturing unit in Sri Lanka is known as Lanka Special Steel, while the joint venture in Thailand for limestone mining is known as Sila Eastern. Recently, Tata Steel commissioned its 1.2 million tonne capacity Cold Rolling Mill complex at 'Global Speed and Cost'. Its fifth phase of the Modernization Programme leverages the intellectual capabilities of its employees to generate sustainable value for the stakeholders. Tata

Steel is taking better Knowledge Management initiatives to shift focus from creating new physical assets to utilizing them with ingenuity and a sturdy business sense.

The Company's community based initiatives far exceeds its business mandate. Its numerous socially responsible activities are aimed at those living in and around its areas of operations, including it's, mines and collieries. The Community Development and Social Welfare, Rural and Tribal Services, Centre for Family Initiatives and Sports departments run and manage programmes which are designed to improve living conditions of the socially and economically under privileged. These are self-sustaining programmes and involve the maximum participation by the target groups. Income generation schemes for women, farmers and youth, safe drinking water in rural areas, health clinics, drugs, alcohol and HIV/AIDS awareness programmes, youth involvement in sports and cultural pursuits are some of the significant activities taken by the Company.

The quest for excellence at Tata Steel is not just a process, but also a way of life. A determination to move up the value chain in process, products and performance has resulted in Tata Steel being acknowledged for its excellence. It was adjudged the Best-Integrated Steel Plant by the Ministry of Steel for 2000-01 and was conferred the Prime Minister's Trophy, for the third time in row and four times in all. The JRD Quality Value Award for business excellence; the CII-EXIM Award for corporate excellence from the CII and EXIM Bank; the Corporate Governance Award, instituted by the Union Finance Ministry for excellence in corporate governance, are testimony to the Steel Company's commitment to excel in all activities it undertakes. Tata Steel has been awarded the Export Engineering Promotion Council Award-2002 and the Tata Steel Website has been Declared Best for 2002 by International Iron & Steel Institute, Belgium in the "IISI People's Choice Award" category. Till date, eight divisions, including its steel works and its mines and collieries have been ISO-14001 certified for environment management. This certification is a reaffirmation of Tata Steel's belief that better environmental management leads to superior business performance.

Tata Steel's concerted effort to become the lowest cost producer of steel has yielded rich dividends in making it a "world class" steel maker and delivering substantial profits. The

intrinsic strength of the company such as low operating costs, special organizational culture and good profitability has been widely appreciated. The intrinsic strength of the company has been widely appreciated and this has led to establishing strategic partnership with such international players as Nippon Steel Corporation, Japan; Arcelor, France, POSDATA, South Korea (a subsidiary of POSCO), Ryerson (a JV with Ryerson Tull of USA), Vivendi Water, UK and Paul Wurth, Luxemborg.

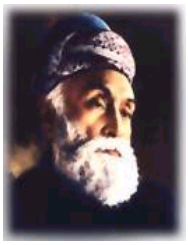
Throughout its history, Tata Steel has adapted itself to face the challenges of time through relentless change. It has continuously been on the growth path and is striving to improve the EVA of the company by seizing the opportunities of tomorrow and exploring newer avenues of operations in Ferro-chrome and titanium and by enhancing capacity of the existing plants.

In January this year, Tata Steel said it would acquire the **Anglo-Dutch steelmaker Corus for $13 billion** in a hotly-contested deal that saw participation of large global steel players. The acquisition has pushed Tata Steel up to the **sixth largest** [2] position globally.

Facts at a Glance

The Founder

Jamshedji Nusserwanji Tata (1839 – 1904)



Jamshedji Nusserwanji Tata [3] ranks among the greatest visionaries of industrial enterprise of all time. Gifted with the most extraordinary imagination and presence, he laid the foundations of Indian Industry, contributed to its consolidation, and became a key figure in India's Industrial renaissance.

The Pioneers

Sir Dorabji Tata (1859-1933)

J.N.Tata had exhorted to his sons to pursue and develop his life's work; his elder son Dorab Tata [4] carried out the bequest with scrupulous zeal and distinction.

Thus even though it was Jamshedji Tata who had envisioned the mammoth projects, it was in fact Dorab Tata who actually brought the ventures to existence and fruition. He was the chairman of the gigantic Tata Enterprise.

Bharat Ratna Jahangir Ratanji Dadabhai Tata (1904-1993)

J.R.Tata [5] has been one of the greatest builders and personalities of modern India in the 20th century. He assumed Chairmanship of Tata Son's Limited at the young age of 34; but his charismatic discipline and forward looking leadership over the next 50 years

And more, led the Tata Group to new heights of achievement, expansions and modernization. Under his stewardship, the number Of Tata venture grew from 13 to around 80, Encompassing Steel, Power generation, Engineering, Hotels, Consultancy Services, Information Technology, Art and Culture, Consumer Goods, Industrial Products etc.

"One must forever strive for excellence or even perfection, in any task however small, and never be satisfied with the second best" **J.R.D. Tata**

Sir Ratan Tata

Jamshedji's younger son [6] had a very distinct personality of his own hat reflected his sensitive understanding of human endeavor. His passion for fine arts was legendary. A philanthropist all his life, he created a trust fund for "the advancement of learning and for the relief of human suffering and other works of public utility".

Awards & Recognitions

1. World Steel Dynamics [7] has ranked Tata Steel as the **world's best steel maker** (for two consecutive years) in its annual listing in February 2006.
2. Tata Steel has been conferred the **Prime Minister of India's Trophy** for the Best Integrated Steel Plant **five times**. [1]
3. It has been awarded **Asia's Most Admired Knowledge Enterprise** award in 2003 and 2004. [1]

Information Technology Services

Information Technology Services (ITS) is as in house IT service provider to all departments of Tata Steel. The department is entrusted with the responsibility of assessing customer requirements, architecting IT solutions, and delivering IT services to Tata Steel. ITS is also responsible for supporting the Information Management process which is one of Tata Steel's kit business processes. ITS has committed itself of create business value for its customers and aspires to take Tata Steel to new levels of excellence through e-transformation.

Tata Steel has been an early adapter of IT. From the batch processing systems of the 1960s, custom-built solution of 1990s on IBM mainframe platforms [8] to the integrated ERP [9] systems of the millennium, Tata Steel has shown how IT can be used to manage business. Acquisition of SAP R/3 competency, introduction of e-business, Data warehouse, Business intelligence and Advance planning optimization solutions are some of the leading edge technologies that ITS has recently brought to Tata Steel. The reach and range of information technology have been significantly expended, with best communication practices.

Department of Research and Development

The establishment of a Research Department, the first of its kind in India, way back in **1935,** stands as a testimony to the spirits of the early pioneers and the vision of the founder of TATA STEEL.

The Company's strength in research and development has consistently helped to meet the challenges of growth and change.

By that time, the First World War was over and at the instance of the Indian Government, TATA STEEL had already **supplied 2,400 km of special Steel Rails and 3 Lakh tons of special grade steel for the manufacture of Combat Helmets, Jerri Cans and other materials for Allied Forces.** Lord Chelmsford, the then Viceroy was highly impressed and for the services rendered by TATA STEEL during the war.

Not that there were no provisions or facilities to carry out laboratory tests for improvement in quality and production standard earlier, but it was only in 1935, when on Nov 6th, the Foundation Stone of the new **R&D Building was laid by Mr. A.R. Dalal**, the then Director of Tata Sons Limited before a distinguished gathering. A new saga began...

Meanwhile in civil front too, **TATA STEEL's R&D made significant contributions for the construction of Howrah Bridge over river Hooghly in Calcutta**. It supplied varieties of low alloy structural steel –TISCORM, Die-steel was supplied for manufacture of small coins, low-tungsten steel for Hacksaws, tungsten carbon steel for taps and cartridge drawings dies, silicon-chrome-tungsten steel for pneumatic tools, chrome-carbon steel for Razor blades and Stainless heat-resistant steel for the Post & Telegraphs. R&D never slept…..

Howrah Bridge

Armoured Vehicle – Used in World War II

After a short while the Second World War began. And again the Government of India turned for supplies of steel to TISCO and as before, the R&D stood the test of time supplying special grade steels like TISCORM and TISCOR to the army both of which bore adequate strength to manufacture Bullet proof Armour Plates and cannon shells for use in the war. Years passed by and TATA STEEL's products found markets around the globe. Today its products are being exported to countries like the United Kingdom, USA, Canada, France, Germany, Korea, Taiwan, Malaysia and China, beside others. Before marketing here too, R&D plays major roles.

Awards

1. **National Research and Development Corporation Award to R&D:** The Research and Development of Tata Steel has been conferred the prestigious "Technology Day Meritorious Invention Award" from National Research and Development Corporation (NRDC). The award was presented to Dr Ananya Mukhopadhyay, Dr Sudipta Sikdar, Mr Ashwin Pandit and Mr Saurabh Kundu of Tata Steel for their extraordinary invention on the development of the "On-line Property Prediction System for Hot Rolled Coils (OPPRESS)". **The joint award carries a cash amount of Rs.1,00,000 and a citation for all the members of the team**

2. **R&D bags 'NIIS Award for R&D Laboratory in Industry' NACE International India Section (NIIS),** a member of NACE International, USA the largest Professional Technical Society in the field of Corrosion has selected Tata Steel R&D Laboratory for the prestigious "NIIS Award for R&D Laboratory in Industry". This award as the Best Laboratory in Industry in the field of Corrosion is being given to Tata Steel R&D in recognition of their significant contributions to Corrosion Awareness in India through research work carried out in corrosion prevention, as well for corrosion evaluation/ testing and training. The award has been recommended by the selection committee based on the infrastructural laboratory facilities availability and its effective utilization, types of tests conducted and the standards adhered to for corrosion evaluation, research work carried out by the lad the lab in corrosion monitoring, and training of personnel in the field of corrosion monitoring. The award was presented at the CORCON 2004 conference in New Delhi on December 2, 2004 by Mr. Anil Deshmukh, Minister and Chairman, Maharashtra State Road Corporation, Govt. of Maharashtra

Corus

Corporate Information

Type: Subsidiary

Founded: Koninklijke Hoogovens N.V. in 1918 [21]

Merger of British Steel & Koninklijke Hoogovens 1999

Headquarters: London, England, UK

Key people: Philippe Varin, CEO. Rauke Henstra, Chief Operating Officer (COO)

Industry: Steel

Products: Steel

Revenue: £10,142 million (2005)

Operating income: £680 million (2005)

Net income: £451 million (2005)

Employees: 50,000

Parent : Tata Group

Corus is Europe's second largest steel producer with annual revenues of over £11 billion and a crude steel production of about 20 million tonnes.

With main steelmaking operations primarily in the UK and the Netherlands, Corus provides innovative solutions to the construction, automotive, packaging, mechanical engineering and other markets worldwide.

Corus comprises three operating Divisions, **Strip Products, Long Products and Distribution & Building Systems** and has a global network of sales offices and service centers, employing around 41,000 people worldwide.

Combining global expertise with local customer service, Corus offers value, reliability and innovation. The Corus brand represents a mark of quality, loyalty and strength.

Corus is a subsidiary of Tata Steel, the world's sixth largest steel producer. With the recent acquisition of Corus, the combined enterprise has over 28 million tonnes of crude steel production capacity and over 82,700 employees across four continents.

Corus people

Corus is proud of its workforce and recognizes the contribution made by its employees. Corus employs some 41,000 employees worldwide.

Manufacturing

Corus has manufacturing operations in many countries with major plants located in the UK, The Netherlands, Germany, France, Norway and Belgium.

From 1 January 2008 a new structural and management reorganization took effect. Corus is now structured into three divisions. These are:

Strip Products Division,

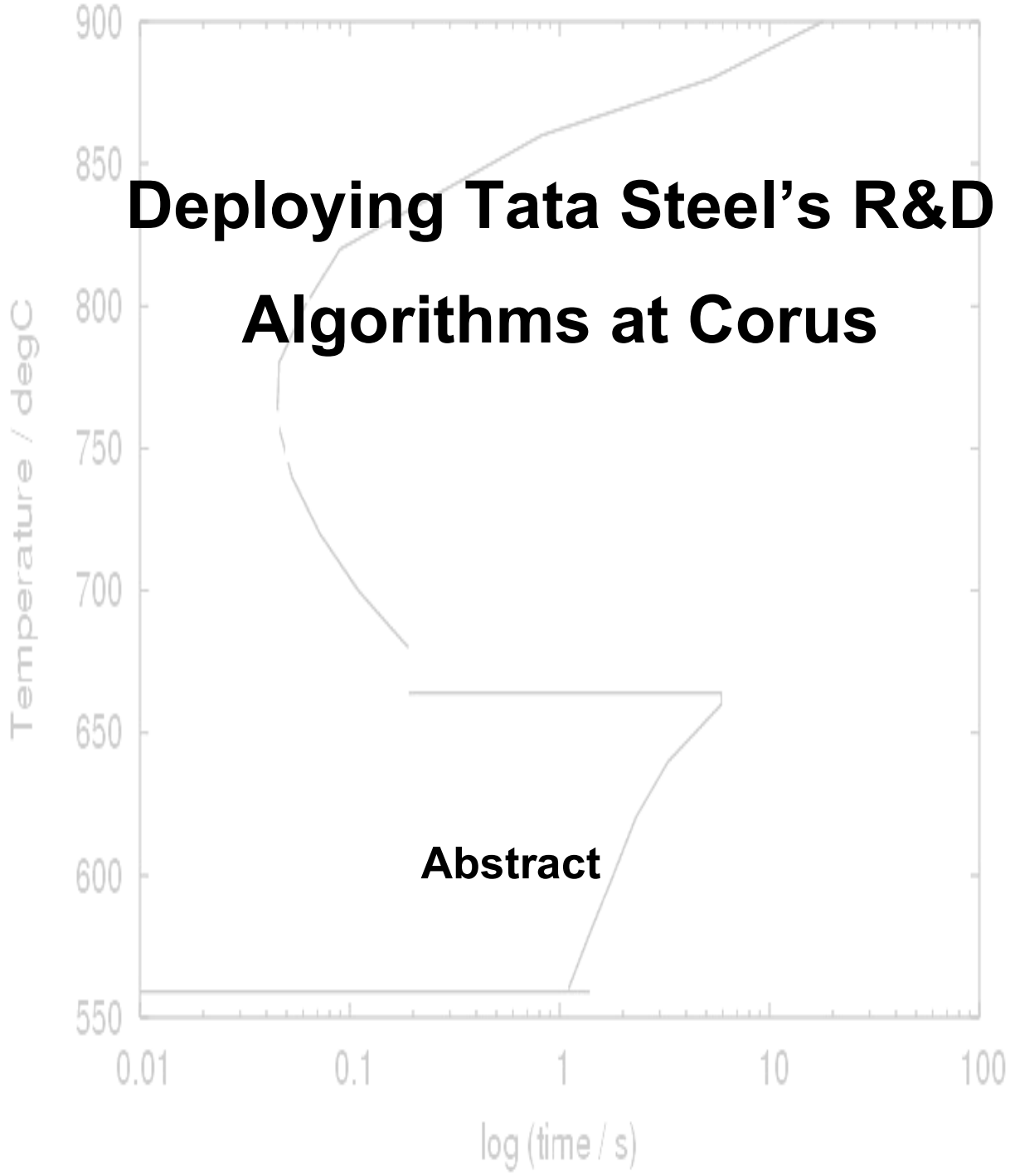
Long Products Division,

Distribution and Building Systems Division

It has major integrated steel plants at Port Talbot in South Wales; Scunthorpe, North Lincolnshire; Teesside, Cleveland (all in the United Kingdom) and IJmuiden in the Netherlands. It also has rolling mills situated at Shotton, North Wales (which manufactures Colorcoat products), Trostre in Llanelli, Llanwern in Newport, South Wales, Rotherham and Stocksbridge, South Yorkshire, England, Motherwell, North Lanarkshire, Scotland, Hayange, France, and Bergen, Norway. In addition it has tube mills located at Corby, Stockton and Hartlepool in England and Oosterhout, Arnhem, Zwijndrecht and Maastricht in the Netherlands. Group turnover for the year to 31 December 2005 was £10.142 billion. Profits were £580 million before tax and £451 million after tax.

On 16 March 2006 Corus announced that it had signed a letter of intent to sell its aluminium rolled products and extrusions businesses to Aleris International, Inc. for £570m. Corus would retain its smelting operations and supply Aleris under a long-term agreement. On 1 August, the sale to Aleris Europe was completed. Primary aluminium production remains with Corus; however, a letter of intent for its sale has been signed.

Corus is the sponsor of the Corus chess tournament in the Netherlands and the UK triathlon team.



Deploying Tata Steel's R&D Algorithms at Corus

Abstract

Section 1 ► Abstract

The project is divided into three modules namely

- **Web Server Setup using Apache** [10]
- **Design and Deployment of a Linux Adapter**
- **CPU Load Balancing in a Linux Cluster by migrating processes** [11] [12]

Proprietary algorithms have been developed over the years at both Tata Steel and Corus [13]. These are the Intellectual Property Rights of the respective companies and hence the internals of these algorithms cannot be exposed. These programs were developed by scientists and researchers in the companies and as such are not built as per software development standards. The apprehension is that these programs may be using idiosyncrasies of the language and the compiler with which these were created leading to portability problems.

The business requirement today is to share these algorithms as generic services that can be consumed by either side.

Distributed computing is all about CPU scheduling and Load Balancing. If there are more than two computers in a cluster, it becomes essential to distribute the CPU load among the processors, so that the task is completed in less time with more throughput.

Distributed computation is fast gaining over single super-computer [14] setup owing to the fact of decreased reliability and increased cost of single super-computing environment. Distributed computing has the advantages of high availability and fault tolerance.

The design of this load balancing system considers one CPU in the cluster as the master CPU and the rest are the clients. The master does the task of database maintenance, which is characterized by CPU and IP information. The Load Balancer [15] does the task of process migration and machine allocation. The system is highly reliable and fault tolerant. It can support CPUs ranging from 1 to 255.

HTTP Server Setup using Apache

Section 2.1 ► Introduction

Section 2.1.1 ► About Apache HTTP Server

- Design by Robert McCool
- Developed by Apache Software Foundation
- Initial release 1996 (11–12 years ago)
- Latest release 2.2.8 / January 19, 2008 (2008-01-19); 103 days ago
- Written in C
- OS Cross-platform
- Available in English
- Genre Web server
- License Apache License 2.0
- Website http://httpd.apache.org/

Section 2.1.2 ► Version Details

Operating System: Fedora Core 4
Kernel Version: 2.6.11-1.1369-FC-4
httpd Version: 2.0.52-19.ent

Section 2.1.3 ► Configuration Details

"LAMP" system: **Linux**, **Apache**, **MySQL** and **PHP**. Depending on what is talked, the **P** also stands for **Perl** or **Python**, but in general, it is assumed to be PHP.

Most out-of-the-box Red Hat Linux installations will have one or more of the LAMP components installed via RPM files.

Log in as `root`

Section 2.1.3.1 ► Configuring MySQL

First, create the group and user that "owns" MySQL. For security purposes, never run MySQL as `root` on the system. To be able to easily identify MySQL processes in `top` or a `ps` list, make a user and group named `mysql`:

```
groupadd mysql
useradd -g mysql -c "MySQL Server" mysql
```

If any error messages about the group or user already existing, that's fine. The goal is just to make sure that the directories are already on the system.

What the `useradd` command is doing is creating a user `mysql` in the group `mysql` with the "name" of MySQL Server. This way when it's showed in various user and process watching apps.

Now change to the "working" directory where the source code is, change the file 'ownership' for the source tree (this prevents build issues in reported in some cases where the packager's username was included on the source.

The `configure` command has many options you can specify. Some fairly common ones are listed here; for help please type:

```
./configure --help | less
```

Type the following commands:

```
cd /usr/local/src/mysql-4.1.22

chown -R root.root *

make clean

./configure \
--prefix=/usr/local/mysql \
--localstatedir=/usr/local/mysql/data \
```

--disable-maintainer-mode \
--with-mysqld-user=mysql \
--with-unix-socket-path=/tmp/mysql.sock \
--without-comment \
--without-debug \
--without-bench

Compile and install the sources using the following commands:

make && make install

Configure MySQL

MySQL is "installed" but few more steps until it's actually "done" and ready to start. First run the script which actually sets up MySQL's internal database (named, oddly enough, `mysql`).

./scripts/mysql_install_db

Then set the proper ownership for the MySQL directories and data files, so that only MySQL (and `root`) can do anything with them.

chown -R root:mysql /usr/local/mysql
chown -R mysql:mysql /usr/local/mysql/data

Copy the default configuration file for the expected size of the database (small, medium, large, huge)

cp support-files/my-medium.cnf /etc/my.cnf
chown root:sys /etc/my.cnf
chmod 644 /etc/my.cnf

If an error message sprouts up about the `data` directory not existing, etc., something went wrong in the `mysql_install_db` step above.

Now inform the system where to find some of the dynamic libraries that MySQL will need to run. The use of dynamic libraries instead of static keep the memory usage of the MySQL program itself to a minimum.

Now do the following:

```
echo "/usr/local/mysql/lib/mysql" >> /etc/ld.so.conf
ldconfig
```

Now create a startup script, which enables MySQL auto-start each time your server is restarted.

```
cp ./support-files/mysql.server /etc/rc.d/init.d/mysql
chmod +x /etc/rc.d/init.d/mysql
/sbin/chkconfig --level 3 mysql on
```

Then set up symlinks for all the MySQL binaries, so they can be run from anyplace without having to include/specify long paths, etc.

```
cd /usr/local/mysql/bin
for file in *; do ln -s /usr/local/mysql/bin/$file /usr/bin/$file; done
```

MySQL Security Issues

First, it is assumed that only applications on the same server will be allowed to access the database (i.e., not a program running on a physically separate server). So configure MySQL not to even listen on port 3306 for TCP connections like it does by default.

Edit `/etc/my.cnf` and uncomment the

```
skip-networking
```

line (delete the leading #).

For more security info, check this great tutorial over at SecurityFocus.

Start MySQL

First, test the linked copy of the startup script in the normal server runlevel start directory, to make sure the symlink was properly set up:

```
cd ~
/etc/rc.d/rc3.d/S90mysql start
```

If you ever want to manually start or stop the MySQL server, use these commands:

```
/etc/rc.d/init.d/mysql start
/etc/rc.d/init.d/mysql stop
```

Let's "test" the install to see what version of MySQL is currently running:

mysqladmin version

It should answer back with the version that has just been installed...

Now we'll set a password for the MySQL `root` user (note that the MySQL `root` user is **not** the same as the system `root` user, and **definitely** should not have the same password as the system `root` user!).

mysqladmin -u root password *new-password*

MySQL is now installed and running on your server.

Test MySQL

To run a quick test, use the command line program `mysql`:

mysql -u root -p

and enter your new `root` user password when prompted. You will then see the MySQL prompt:

mysql>

First, take care of another security issue and delete the sample database `test` and all default accounts except for the MySQL root user. Enter each of these lines at the `mysql>` prompt:

drop database test;
use mysql;
delete from db;
delete from user where not (host="localhost" and user="root");
flush privileges;

As another security measure, change the MySQL administrator account name from `root` to something harder to guess. This will make it that much harder for someone who gains shell access to the server to take control of MySQL.

update user set user="sqladmin" where user="root";
flush privileges;

Now, on with the "standard" testing... First, create a new database:

create database foo;

The following results should be seen:

Query OK, 1 row affected (0.04 sec)

mysql>

Delete the database:

drop database foo;

You should see the result:

Query OK, 0 rows affected (0.06 sec)

mysql>

To exit from `mysql` enter `\q`:

\q

Section: 2.1.3.2 ► Build and Install Apache

The advantage to building Apache with support for dynamically loaded modules is that in the future, one can add functionality to the webserver by just compiling and installing modules, and restarting the webserver. If the features were compiled into Apache, one would need to rebuild Apache from scratch every time you wanted to add or update a module (like PHP). The Apache binary is also smaller, which means more efficient memory usage.

The downside to dynamic modules is a slight performance hit compared to having the modules compiled in.

cd /usr/local/src/apache_1.3.37

make clean

```
./configure \
--prefix=/usr/local/apache \
--enable-shared=max \
--enable-module=rewrite \
--enable-module=so

make && make install
```

Section 2.3.1.3 ► Build and Install PHP

This section has only been tested with PHP v4.x. Please note that there are **many** options which can be selected when compiling PHP. Some will have library dependencies, meaning certain software may need to be already installed on your server before you start building PHP. Use the command

```
./configure --help | less
```

once you change into the PHP source directory. This will show you a list of all possible configuration switches.

```
cd /usr/local/src/php-4.4.6

./configure \
--with-apxs=/usr/local/apache/bin/apxs \
--disable-debug \
--enable-ftp \
--enable-inline-optimization \
--enable-magic-quotes \
--enable-mbstring \
--enable-mm=shared \
--enable-safe-mode \
--enable-track-vars \
```

--enable-trans-sid \

--enable-wddx=shared \

--enable-xml \

--with-dom \

--with-gd \

--with-gettext \

--with-mysql=/usr/local/mysql \

--with-regex=system \

--with-xml \

--with-zlib-dir=/usr/lib

make && make install

cp php.ini-dist /usr/local/lib/php.ini

Keep the config files all together in `/etc`. Set up a symbolic link like this:

ln -s /usr/local/lib/php.ini /etc/php.ini

Then I can just open `/etc/php.ini` in the editor to make changes.

Edit the Apache Configuration File (`httpd.conf`)

Keep all the configuration files together in `/etc`, so set up a symbolic link from the actual location to `/etc`:

ln -s /usr/local/apache/conf/httpd.conf /etc/httpd.conf

Now open `/etc/httpd.conf` in your favorite text editor, and set all the basic Apache options in accordance with the official Apache instructions

To ensure your PHP files are properly interpreted, and not just downloaded as text files, remove the # at the beginning of the lines which read:

#AddType application/x-httpd-php .php

#AddType application/x-httpd-php-source .phps

If the AddType lines above don't exist, manually enter them (without the leading # of course) after the line

AddType application/x-tar .tgz

or anyplace within the `<IfModule mod_mime.c>` section of `httpd.conf`.

If you wish to use other/additional extensions/filetypes for your PHP scripts instead of just `.php`, add them to the `AddType` directive:

AddType application/x-httpd-php .php **.foo**
AddType application/x-httpd-php-source .phps **.phtmls**

An example: if you wanted every single HTML page to be parsed and processed like a PHP script, just add `.htm` and `.html`:

AddType application/x-httpd-php .php **.htm .html**

There will be a bit of a performance loss if every single HTML page is being checked for PHP code even if it doesn't contain any. But if you want to use PHP but be "stealthy" about it, you can use this trick.

Add `index.php` to the list of valid Directory Index files so that your "default page" in a directory can be named `index.php`.

<IfModule mod_dir.c>
DirectoryIndex **index.php** index.htm index.html
</IfModule>

You can add anything else you want here too. If you want `foobar.baz` to be a valid directory index page, just add the `.baz` filetype to the `AddType` line, and add `foobar.baz` to the `DirectoryIndex` line.

Start Apache

We want to set Apache up with a normal start/stop script in `/etc/rc.d/init.d` so it can be auto-started and controlled like other system daemons. Set up a symbolic link for the `apachectl` utility (installed automatically as part of Apache):

ln -s /usr/local/apache/bin/apachectl /etc/rc.d/init.d/apache

Then set up auto-start for runlevel 3 (where the server will go by default):

ln -s /etc/rc.d/init.d/apache /etc/rc.d/rc3.d/S90apache

Then start the daemon:

/etc/rc.d/init.d/apache start

You can check that it's running properly by doing:

ps -ef

and look for the `httpd` processes.

Service Oriented Architecture (S.O.A.)

Section 3.1►Phase 1 ► Service Oriented Architecture – Introduction

Service Oriented Architecture (SOA) [16] is a computer systems architectural style for creating and using business processes, packaged as services, throughout their lifecycle. SOA also defines and provisions the IT infrastructure to allow different applications to exchange data and participate in business processes. These functions are loosely coupled with the operating systems and programming languages underlying the applications. SOA separates functions into distinct units (services), which can be distributed over a network and can be combined and reused to create business applications. These services communicate with each other by passing data from one service to another, or by coordinating an activity between two or more services. SOA concepts are often seen as built upon and evolving from older concepts of distributed computing and modular programming.

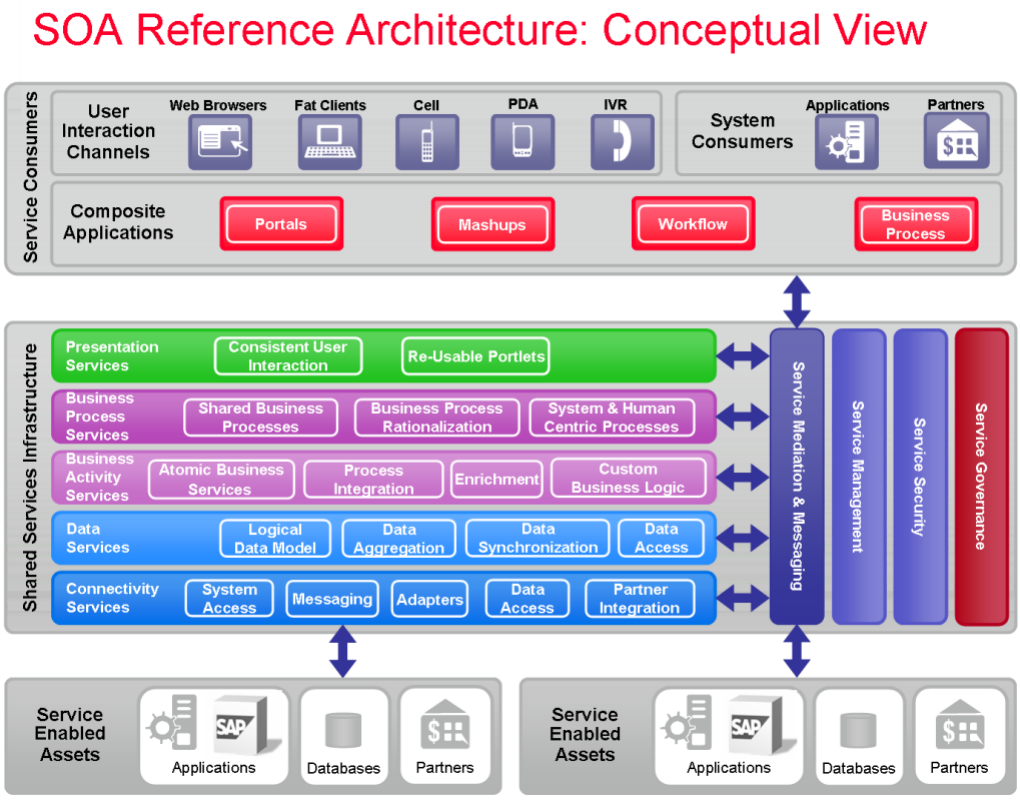


Figure 1: BEA's approach to SOA

Architecture is not tied to a specific technology. It may be implemented using a wide range of technologies, including **SOAP, REST, RPC, DCOM, CORBA, Web Services or WCF**. SOA can be implemented using one or more of these protocols and, for example, might use a file system mechanism to communicate data conforming to a defined interface specification between processes conforming to the SOA concept. The key is independent services with defined interfaces that can be called to perform their tasks in a standard way, without the service having foreknowledge of the calling application, and without the application having or needing knowledge of how the service actually performs its tasks.

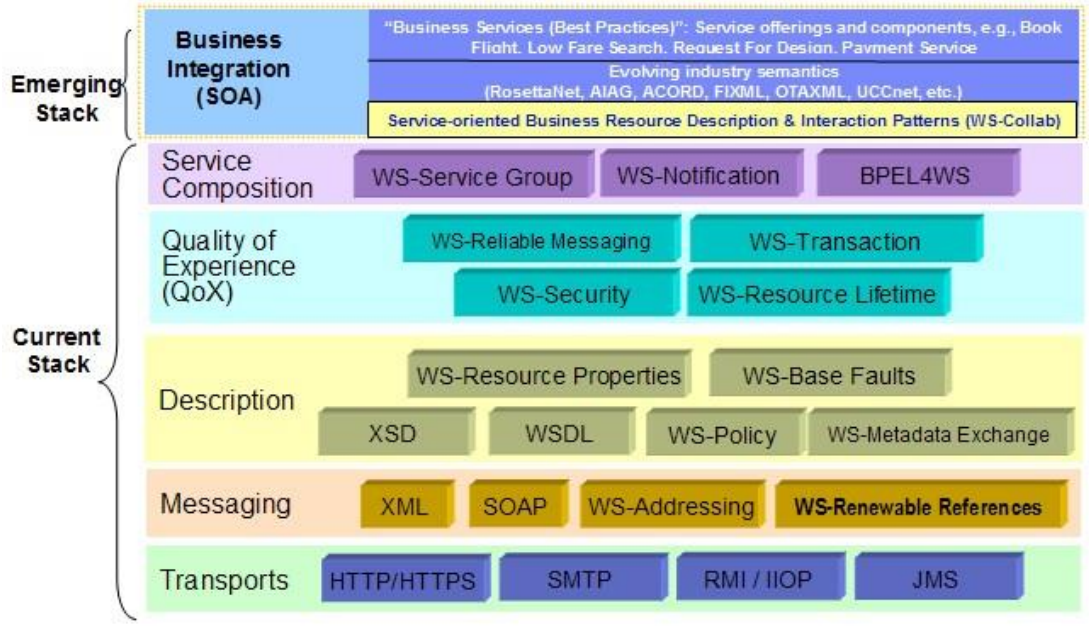


Figure 2: IBM's Model of SOA

The following guiding principles define the ground rules for development, maintenance, and usage of the SOA:

Reuse, granularity, modularity, composability, componentization, and interoperability Compliance to standards (both common and industry-specific) Services identification and categorization, provisioning and delivery, and monitoring and tracking

The following specific architectural principles for design and service definition focus on specific themes that influence the intrinsic behaviour of a system and the style of its design:

- **Service encapsulation** - Many web-services are consolidated to be used under the SOA Architecture. Often such services have not been planned to be under SOA.
- **Service loose coupling** - Services maintain a relationship that minimizes dependencies and only requires that they maintain an awareness of each other
- **Service contract** - Services adhere to a communications agreement, as defined collectively by one or more service description documents
- **Service abstraction** - Beyond what is described in the service contract, services hide logic from the outside world
- **Service reusability** - Logic is divided into services with the intention of promoting reuse
- **Service composability** - Collections of services can be coordinated and assembled to form composite services
- **Service autonomy** – Services have control over the logic they encapsulate
- **Service optimization** – All else equal, high-quality services are generally considered preferable to low-quality ones
- **Service discoverability** – Services are designed to be outwardly descriptive so that they can be found and assessed via available discovery mechanisms.

The following references provide additional considerations for defining a SOA implementation:

In addition, the following factors should be taken into account when defining a SOA implementation:

- **Efficient use of system resources**
- **Service maturity and performance**
- **EAI Enterprise Application Integration**

Challenges faced in SOA adoption

One obvious and common challenge faced is managing services metadata. SOA-based environments can include many services, which exchange messages to perform tasks. Depending on the design, a single application may generate millions of messages. Managing and providing information on how services interact is a complicated task.

Another challenge is providing appropriate levels of security. Security model built into an application may no longer be appropriate when the capabilities of the application are exposed as services that can be used by other applications. That is, application-managed security is not the right model for securing services. A number of new technologies and standards are emerging to provide more appropriate models for security in SOA.

As SOA and the WS-* specifications are constantly being expanded, updated and refined, there is a shortage of skilled people to work on SOA based systems, including the integration of services and construction of services infrastructure.

Interoperability is another important aspect in the SOA implementations. The WS-I organization has developed Basic Profile (BP) and Basic Security Profile (BSP) to enforce compatibility. Testing tools have been designed by WS-I to help assess whether web services are conformant with WS-I profile guidelines. Additionally, another Charter has been established to work on the Reliable Secure Profile.

There is significant vendor hype concerning SOA that can create expectations that may not be fulfilled. Product stacks are still evolving as early adopters test the development and runtime products with real world problems. SOA does not guarantee reduced IT costs, improved systems agility or faster time to market. Successful SOA implementations may realize some or all of these benefits depending on the quality and relevance of the system architecture and design.

SOA and Business Architecture

One area where SOA has been gaining ground is in its power as a mechanism for defining business services and operating models and thus provide a structure for IT to deliver against the actual business requirements and adapt in a similar way to the business. The purpose of using SOA as a business mapping tool is to ensure that the services created properly represent the business view and are not just what technologists think the business services should be. At the heart of SOA planning is the process of defining architectures for the use of information in support of the business, and the plan for implementing those architectures. Enterprise Business Architecture should always represent the highest and most dominant architecture. Every service

should be created with the intent to bring value to the business in some way and must be traceable back to the business architecture.

Within this area, SOMA (service-oriented modeling and architecture) was announced by IBM as the first publicly announced SOA-related methodology in 2004. Since then, efforts have been made to move towards greater standardization and the involvement of business objectives, particularly within the OASIS standards group and specifically the SOA Adoption Blueprints group. All of these approaches take a fundamentally structured approach to SOA, focusing more on the Services and Architecture elements and leaving implementation to the more technically focused standards. Another pertinent example is SAP Enterprise Services Architecture, which is focused on a strict governance process and the use of semantics to improve the usefulness of services in business process innovation.

Section 3.2 ►Phase 1 ► Web Services

Web services [19] are software systems that make it easier for different systems to communicate with one another automatically in order to pass information or conduct transactions.

Characterized by:

- Small functions built up to a bigger function
- Disaggregate larger systems and distribute
- Can be distributed anywhere, plugged in anywhere
- Loose coupling
- Evolving into a software/functionality pick and mix.

Analogy: Lego

Figure 3: Web Services Technologies

Examples:

- Microsoft Passport
- Microsoft Live Search Web service
- Amazon Simple Storage Service (Amazon S3)
- Amazon Elastic Compute Cloud (Amazon EC2) – Beta
- Google web services - search, spell check etc.

Pros

- A Web Service is accessible over the Web
- Web Services are platform-independent and language-neutral (HTTP+XML)
- Build Internet-scale application
- Can be used across Internet proxies and firewalls
- A Web Service provides an interface that can be called from another program.
- A Web Service is registered and can be located through a Web Service Registry.
- Web Services support loosely coupled connections between systems.

Cons

- Overhead. Transmitting all your data in XML is not as efficient as using a proprietary binary code.
- The object state may not be transmitted in entirety.
- Because of overhead cannot be used in critical real-time applications

Web services in the.Net Stack

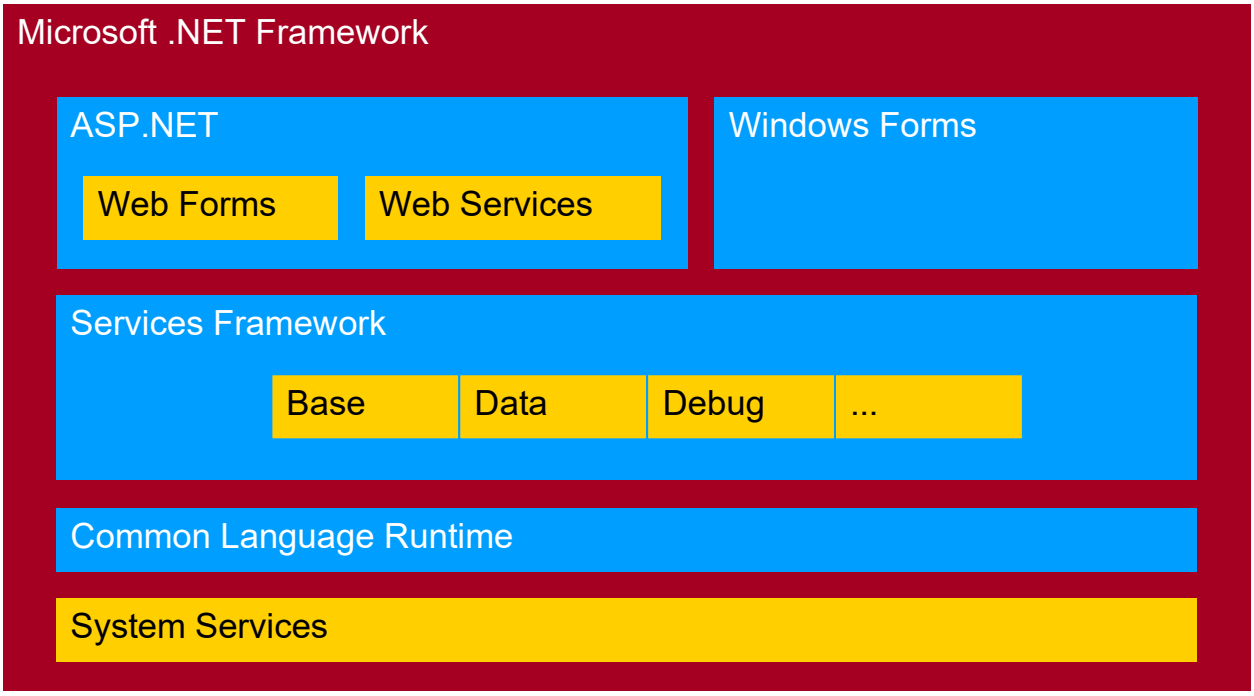


Figure 4: Web Services Defined in .net Stack

ASP.NET - ASP.NET is a unified Web development model that includes the services necessary for you to build enterprise-class Web applications with a minimum of coding. ASP.NET is part of the .NET Framework, and when coding ASP.NET applications you have access to classes in the .NET Framework. You can code your applications in any language compatible with the common language runtime (CLR), including Microsoft Visual Basic, C#, JScript .NET, and J#. These languages enable you to develop ASP.NET applications that benefit from the common language runtime, type safety, inheritance, and so on.

ASP.NET includes:

- A page and controls framework
- The ASP.NET compiler
- Security infrastructure
- State-management facilities

- Application configuration
- Health monitoring and performance features
- Debugging support
- An XML Web services framework
- Extensible hosting environment and application life cycle management
- An extensible designer environment

Common Language Runtime - The Common Language Runtime (CLR) is the virtual machine component of Microsoft's .NET initiative. It is Microsoft's implementation of the Common Language Infrastructure (CLI) standard, which defines an execution environment for program code. The CLR runs a form of byte code called the Common Intermediate Language (CIL, previously known as MSIL -- Microsoft Intermediate Language).

Developers using the CLR write code in a language such as C# or VB.Net. At compile time, a .NET compiler converts such code into CIL code. At runtime, the CLR's just-in-time compiler (JIT compiler) converts the CIL code into code native to the operating system. Alternatively, the CIL code can be compiled to native code in a separate step prior to runtime. This speeds up all later runs of the software as the CIL-to-native compilation is no longer necessary.

Web services Application Model

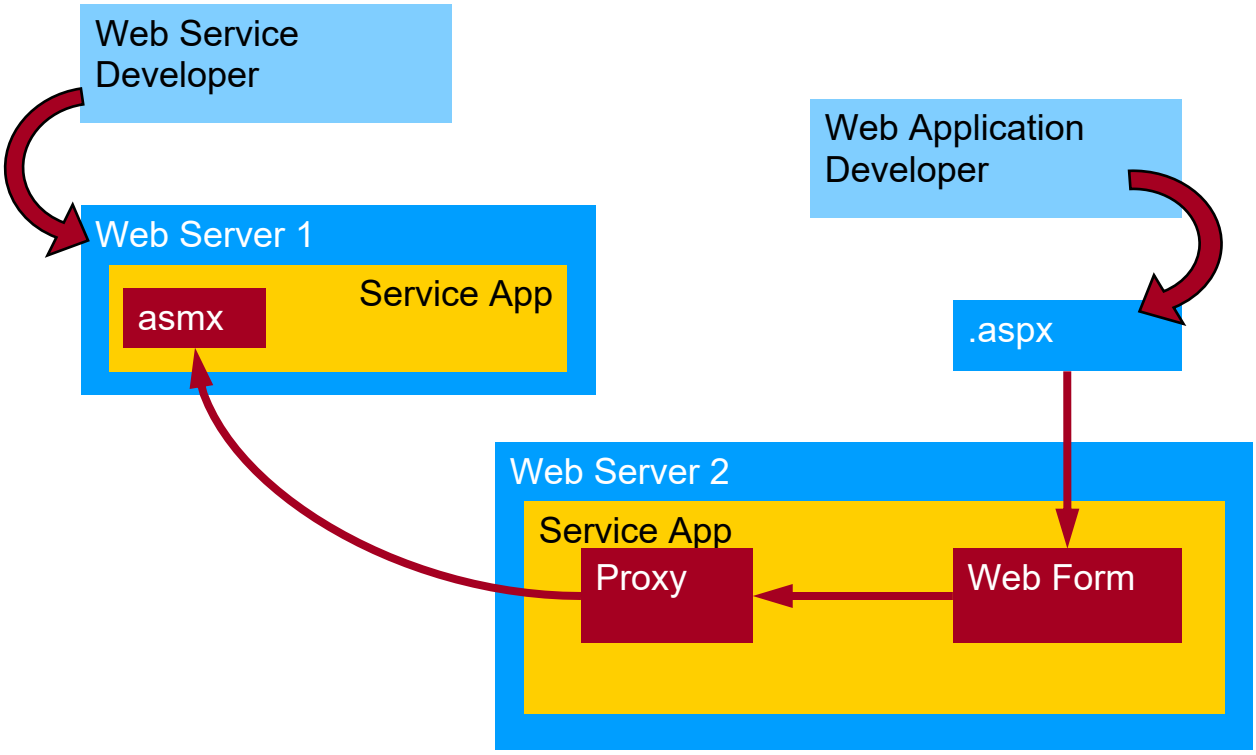


Figure 5: Web Service Application Model

Figure 6: Example of a Web Service

Section 3.3 ►Phase 1 ► Simple Object Access Protocol (S.O.A.P.)

SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses. SOAP can potentially be used in combination with a variety of other protocols; however, the only bindings defined in this document describe how to use SOAP in combination with HTTP and HTTP Extension Framework.

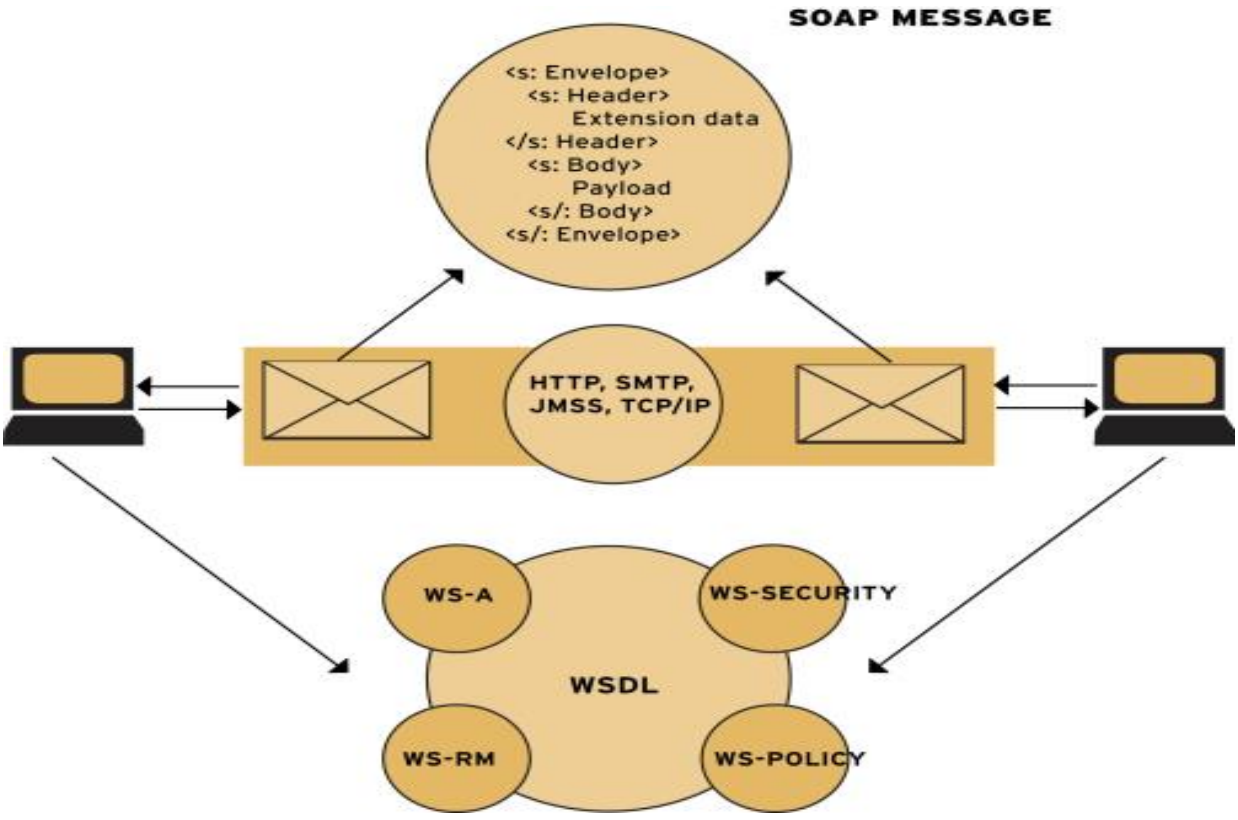


Figure 7: S.O.A.P. Message Format

SOAP Message Examples

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
>
   <SOAP-ENV:Body>
       <m:GetLastTradePrice xmlns:m="Some-URI">
           <symbol>DIS</symbol>
       </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example 1: SOAP Message Embedded in HTTP Request

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
/>
   <SOAP-ENV:Body>
       <m:GetLastTradePriceResponse xmlns:m="Some-URI">
           <Price>34.5</Price>
       </m:GetLastTradePriceResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example 2: SOAP Message Embedded in HTTP Response

Section: 3.4 ► Phase 1 ► Web Services Description Language (W.S.D.L.)

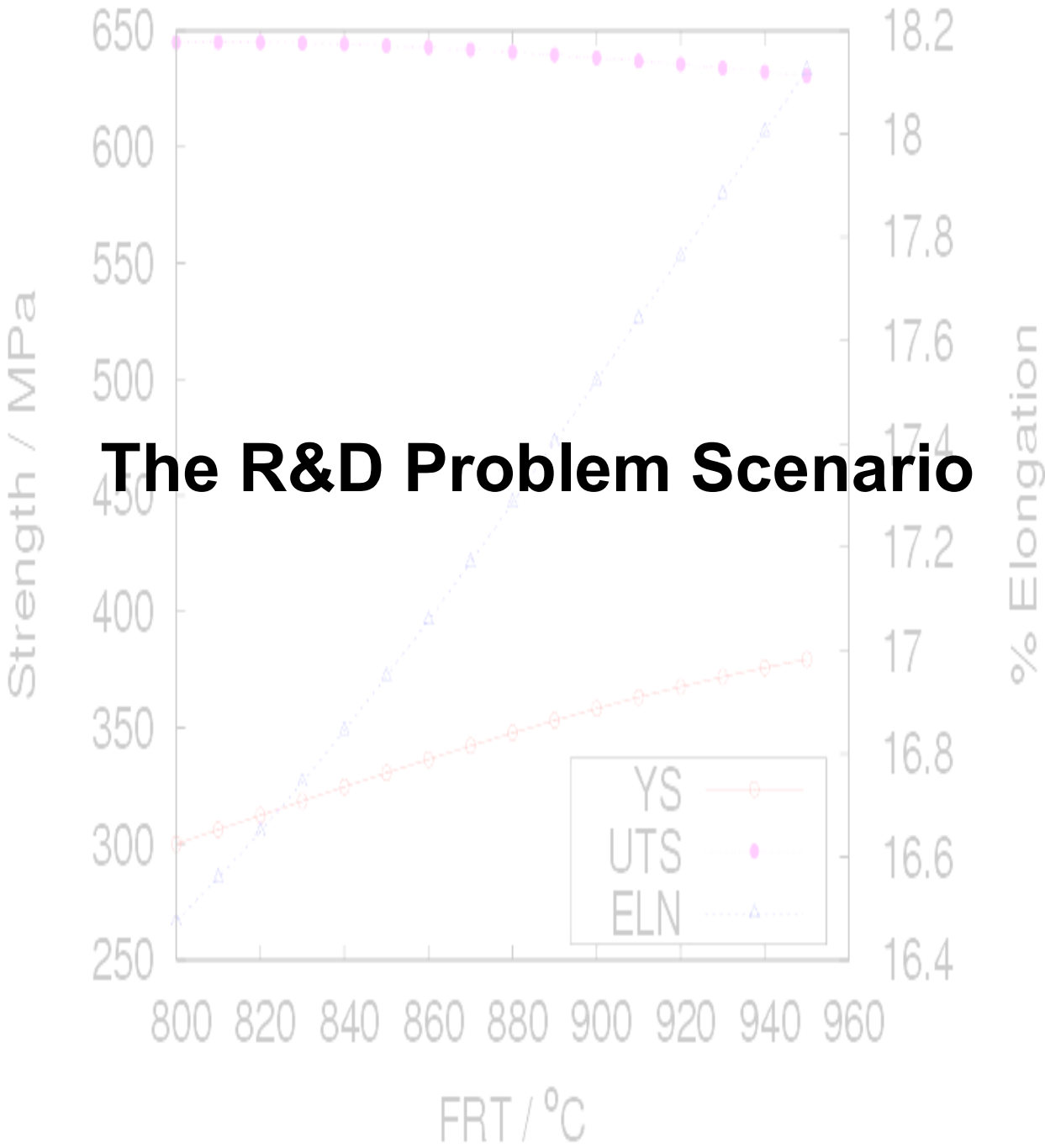
Web Services Description Language (WSDL) message exchange patterns define the sequence and cardinality of abstract messages listed in an operation. Message exchange patterns also define which other nodes send messages to, and receive messages from, the service implementing the operation.

By design, WSDL message exchange patterns abstract out specific message types. Patterns identify placeholders for messages, and placeholders are associated with specific message types by the operation using the pattern.

Unless explicitly stated otherwise, WSDL message exchange patterns also abstract out binding-specific information like timing between messages, whether the pattern is synchronous or asynchronous, and whether the message are sent over a single or multiple channels.

Like interfaces and operations, WSDL message exchange patterns do not exhaustively describe the set of messages exchanged between a service and other nodes; by some prior agreement, another node and/or the service may send other messages (to each other or to other nodes) that are not described by the pattern. For instance, even though a pattern may define a single message sent from a service to one other node, the Web Service may multicast that message to other nodes.

To maximize reuse, WSDL message exchange patterns identify a minimal contract between other parties and Web Services, and contain only information that is relevant to both the Web Service and another party.



The R&D Problem Scenario

Section 3.1 ► Phase 2

Problem Statement: Proprietary algorithms have been developed over the years at both Tata Steel and Corus. These are the Intellectual Property Rights of the respective companies and hence the internals of these algorithms cannot be exposed. These programs were developed by scientists and researchers in the companies and as such are not built as per software development standards. The apprehension is that these programs may be using idiosyncrasies of the language and the compiler with which these were created leading to portability problems.

The algorithms deal with modeling calculations, which are used extensively in the field of research in metallurgy. These algorithms are owned by the Materials Modeling and Product Design Group of Tata Steel Limited. Some of the algorithms at Tata Steel end include:

- Mechanical properties for IFHS,DP,D and EDD grades
- Mill Load for all six stands of HSM
- Time-Temperature-Transformation diagrams
- Micro alloyed steel
- Thickness reduction and crown during rolling

The programs were developed using the following:

- **Programming Languages** :
    1. Corus: Delphi
    2. Tata Steel: C, Fortran, Abacus and PHP

- **Platform**: MS Windows/DOS and GNU/Linux

- **Compilers** : GNU Fortran or C and Visual Studio based

The business requirement today is to share these algorithms as generic services that can be consumed by either side.

Section 3.2 ► Phase 2

Solution Architecture – Proposed by Microsoft® Corporation

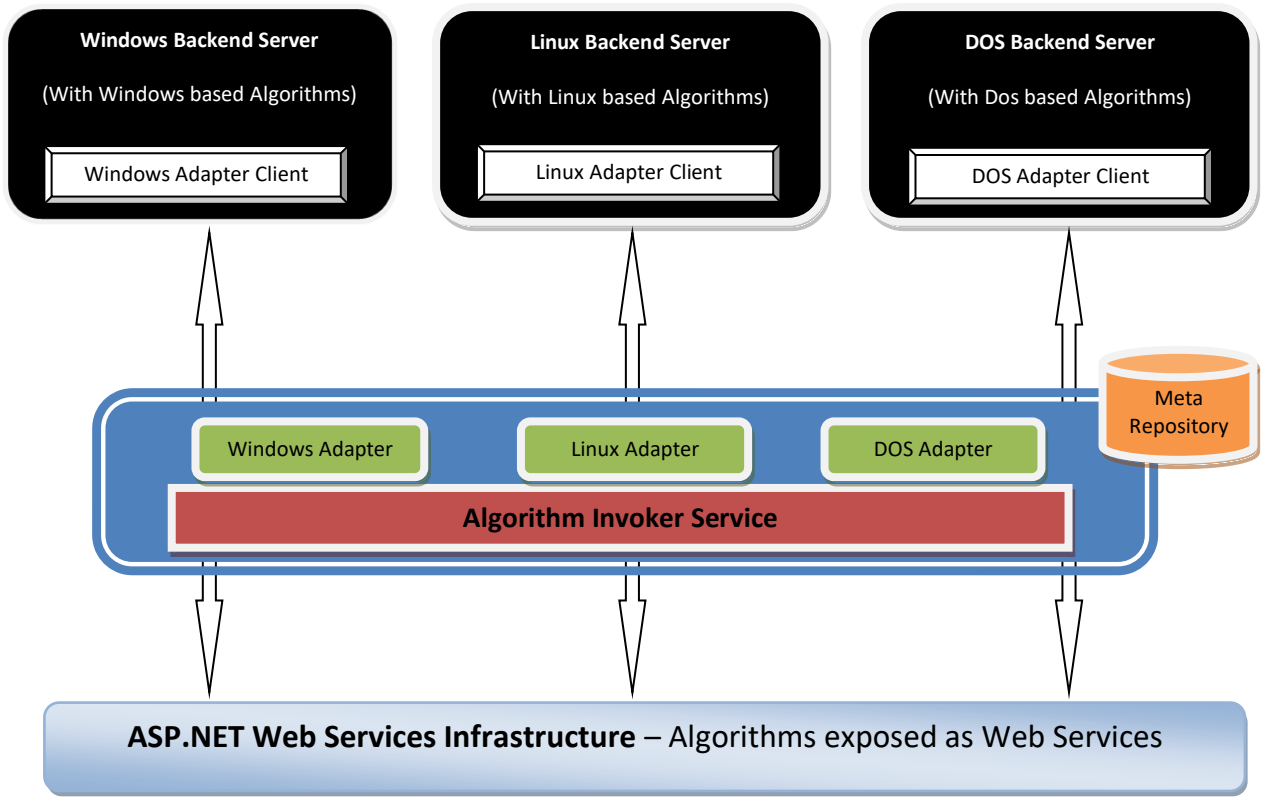


Figure 8: Design Architecture

Description

Windows Backend Server – This is regular Server(s) setup that houses the custom Algorithms on Windows. This server would also have a custom built Windows Adapter Client, which would Invoke the algorithms locally on demand from the Adapter Link Server. This Adapter Client can be implemented as a service on .NET, which communicates with the Adapter Link Server for getting/returning Input/output of the Algorithms. Backend Servers for other environments like Linux, DOS, shall have their own specific adapter client implementations (maybe built using Sockets)

Adapter Link Server – This Server would accept the requests for invoking algorithms from front ending Web Services, and call upon the corresponding Adapter Clients on the Backend Servers. In this Server, there would be different Adapters for communicating with corresponding Adapter Clients for different platforms on the Backend Servers. This server would also house a repository containing mapping between the Algorithms and their corresponding execution environment (Win/Linux/DOS etc), location on the backend server, corresponding adapter client etc.

The request for Invocation of Algorithms, along with any corresponding input arguments required, would come from the front ending ASP.NET web service. This request will be received by the Algorithm Invoker Service, which in turn shall look for the correct algorithm mapping in the Meta repository and call the corresponding Adapter to forward the request to the appropriate Backend Server. This tier controls all communication back and forth and keeps both the front ending tier and the backend server tier, invisible from each other.

ASP.NET Web Services Infrastructure – This tier will house all the web service representations of the corresponding Algorithms. This tier shall coordinate with the Adapter Link Server for Algorithm invocation. Each Web Service in this tier shall correspond to a particular Algorithm on the Backend Servers.

Section 3.3 ► Phase 2 ► Proof of Concept

1. User fills up a form and submits
2. The values are passed onto the algorithm at R&D.
3. The algorithm processes and returns the result

Section 3.3.1 ► Challenges

a) Should map to the solution architecture - Generic
b) The consumer, the web service host and algorithm are on separate systems
c) Consumer and web service host run on Windows 2003 .The algorithm runs on a Linux Cluster @ R&D.
d) The output is a graph plot – a JPEG image
e) Communication among systems to be on TCP/IP stack *(cannot use Disk or File system I/O)*

The Team

Programmers and Brainstormers:

Dr. M Murugananth (R&D), Mr. Fredi Zarolia
Mr. K Krishna Raju and Shoaib Jameel

Microsoft Corporation India
Mr. Bhasker Joshi and Mr. Arjun Bahree

Advisors:
Mr. Indraneel Mazumder and Mr. Ashutosh Kumar

Section 3.4 ► Phase 2 ► Design of the Linux Adapter

The Computing Setup

1. Operating System: Fedora Core 4
2. Linux Kernel Version: 2.6.11-1.1369-FC-4
3. RAM: 256 MB
4. HDD: 80 GB
5. Processor: Intel Pentium 4
6. Processor Cycle Speed: 2.9 GHz.
7. Programming Language: C

Section 3.4.1 ► Solution Strategies ► Strategy 1 ► Simulating the R&D Algorithms

Section 3.4.1.1 ► Problem statement

A remote machine or a web service consumer connects to the machine hosting the web service. The web service-hosting machine, as it receives the client's request along with the set of values, generates a text file in the Input directory, which has to be read and processed by the target Linux machine. The results thus processed have to be delivered back to the windows machine, which will finally publish the results along with an image generated by the modeling algorithm at R&D.

Section 3.4.1.2 ► The Setup

There is a Microsoft Windows machine, running Windows 2003 Server. This machine has a shared directory, which has three sub directories. The Linux box mounts these remote directories using Samba file sharing and does the requisite processing subsequently.

Directory 1: Input

Directory 2: Output

Directory 3: Moved

Input Directory: This directory always remains empty until and unless the web service creates an Input text file with a unique identifier name. All the text files follow one naming pattern, which is characterized by *date, time, minutes, seconds, milliseconds* as their names. Finally, *_Input.txt* is appended to the names of the text files. This file creation and naming is done by the Windows based web service. The Linux Adapter *continuously* scans this Input directory for the arrival of new text files.

Output Directory: The Linux Adapter, the moment it finds the text file in the Input directory, copies the contents of the Inputs to a newly created text file with the same timestamp value and adds an extension of *_Output.txt.* Meanwhile, the adapter subsequently copies a sample image stored in the local storage with the same unique timestamp value and extension *_Output.jpg*. The web service, which was continuously refreshing at regular interval in an Output page, displays the image from the Output directory.

Moved Directory: This directory contains the contents of the Input directory, which is moved by the adapter. The adapter also copies the sample image to the moved directory.

Section 3.4.1.3 ► The Linux Adapter Execution Details

The Linux end has a program (written in C) executing continuously. It monitors the contents of the directory **Input** for the arrival of new file with the extension of *.txt*. As soon as the file arrives, the Linux program reads the file contents and parses the name of the file to segregate the unique name or the time stamp value that is associated with the file name.

Using this unique timestamp value from the file name, the Linux program generates the corresponding Output file and an image.

The windows hosted web service reads the contents of the Output directory at regular intervals to locate the image with the corresponding time-stamp value, the value with which it had created the Input text file.

Diagrammatic Illustration

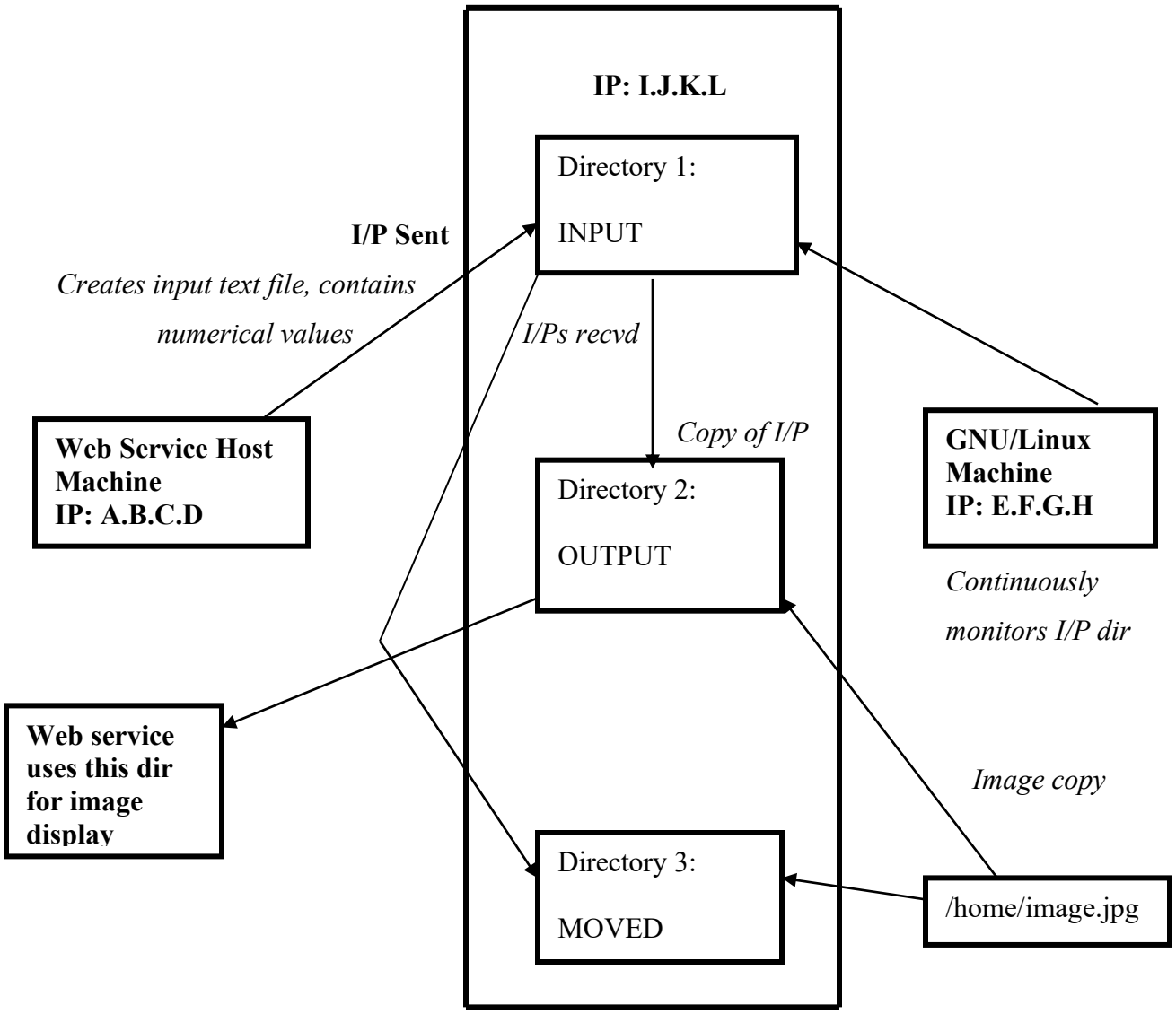


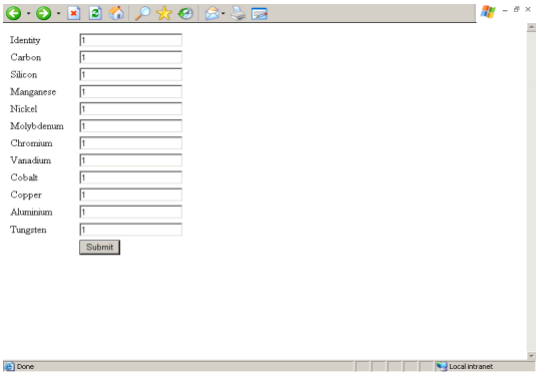
Figure 9: Control Flow

Section 3.4.1.4 ► Explanation

Consider the following scenario:

- Web service creates a text file with the following name: *3042008925520_Input.txt* in the Input directory
- Linux Adapter continuously executes and scans the Input directory for a new text file. When the text file is created, the adapter does the following:
    1. Parses the name of the text file to segregate the unique time stamp value example *3042008925520*
    2. Creates another text file of the name *3042008925520_Output.txt* with similar contents and stores this text file in the Output directory.
    3. Copies an image from the local drive and names it *3042008925520_Output.jpg.* The image is copied to the Output directory.
    4. Moves the *3042008925520_Input.txt* from the Input directory. This input file is moved to the Moved directory. An image with the name 3042008925520_Output.jpg is also moved in the Moved directory.
    5. The web service, which was waiting for the above operation to complete, gets the image from the Output directory and displays it to the user.
    6. A log file is generated for the entire operation.
- The Linux Adapter keeps on executing until infinity.

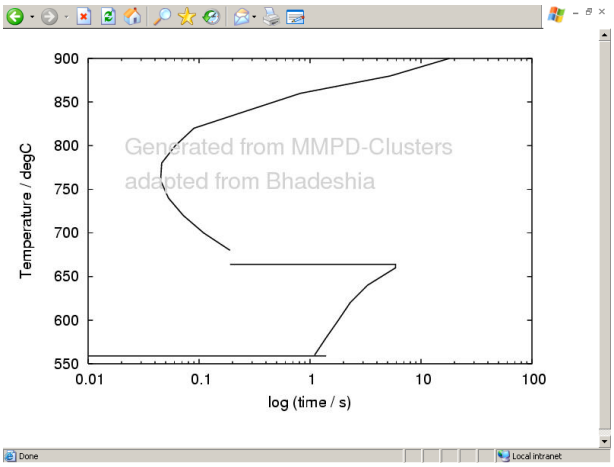
An Illustration of the Implementation



Screen 1: User Gives the Input

Please wait .....

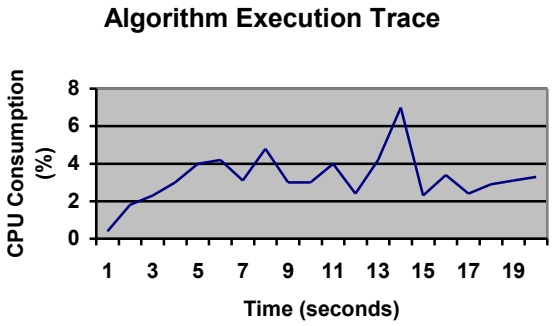
Screen 2: Displays the message while the Adapter is executing



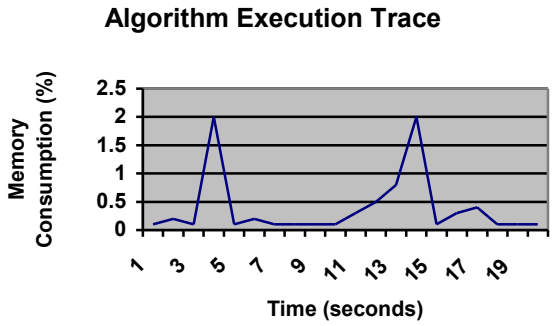
Screen 3: Image Displayed

Section 3.4.1.5 ► Why this implementation failed?

1. The Input directory, if it contained any stale text files, the Linux adapter failed to produce results. It used to crash.
2. The adapter was continuously executing, devouring an appreciable amount of memory and CPU usage. Network bandwidth consumption was another bother. The process i.e. the adapter consumed about **2-3%** of the entire CPU and hogged about **.2%** of the entire memory.
3. It failed to handle multiple requests.
4. **Technical Details:**



Plot 1: CPU Consumption v/s Time



Plot 2: Memory Consumption v/s Time

Section 3.4.1.6 ► Lessons Learnt

1. It makes no sense to design a computer program that runs continuously in an infinite loop. It not only consumes precious CPU resources but also consumes lot of bandwidth, as in this case.
2. If at all it is required to design a program that should continuously execute, the best option is to design and code a **Daemon (Disk and Execution Monitor Program)** that is memory resident. It sleeps when no event takes place and wakes up when an event occurs. This requires an in-depth knowledge of kernel level programming.

Section 3.4.1.7 ►Implementation Details ► Pseudo Code

Step 1 BEGIN

Step 2 Initialize the character array to NULL

Step 3 LOOP Infinitely

Step 4 READ the directory contents and STORE the contents in a file

Step 5 OPEN the file in READ mode

Step 6 READ the contents of the file in a CHAR array

Step 7 IF file IS NOT EMPTY GOTO Step 9

Step 8 ELSE GOTO Step 6

Step 9 STORE the name of the file in a CHAR array

Step 10 Parse the name of the file to extract the unique time stamp value

Step 11 CREATE another text file containing the same contents as the original text file with the file name *unique_timestamp_Output.txt* and STORE the file in the Output directory of the remote Windows share.

Step 12 Copy an image from the local drive and give it the name *unique_timestamp_Ouput.jpg* and STORE it in the Output directory of the remote Windows share.

Step 13 Copy the same image file in Moved Directory

Step 14 Move the original text file to the Moved directory

Step 15 CREATE a log file for the entire operation

Step 16 FREE all the array dynamically allocated

Step 17 GOTO Step 3

Section 3.4.2 ► Solution Strategies ► Strategy 2 ► Adapter Implementation Using sockets (TCP/IP) with Disk I/O

The Computing Setup

1. Operating System: Fedora Core 4
2. Linux Kernel Version: 2.6.11-1.1369-FC-4
3. RAM: 256 MB
4. HDD: 80 GB
5. Processor: Intel Pentium 4
6. Processor Cycle Speed: 2.9 GHz.
7. Programming Language: C
8. Compiler – GNU's gcc ver – 4.0.0

Process Runtime Statistics

ps –aux | grep *file_name.out* gave

CPU Usage = 0.0% when in listening mode
Memory Usage = 0.0% (in listening mode)

When the Adapter executed

top command showed:

CPU (%) = 0.3
Memory (%) = 0.1

The adapter continuously listened a particular port on the Linux machine for any incoming requests. This adapter used the BSD's socket API.

When a user fills in the form with the relevant data on *IP:A.B.C.D/client* web page, he/she is redirected to the *IP:A.B.C.D/output.aspx* web page where he/she waits for the result to

be displayed. The web page *refreshes every 5 seconds*. When the Output file arrives in the Output directory the results are instantly displayed as an image.

Meanwhile, as the user is waiting for the Output a lot goes on in the background. The steps below give the background details of what actually happens.

Step 1: User clicks Submit on *IP:A.B.C.D/client*

Step 2: There is a program at the Windows end that creates an Input file in the Input directory simultaneously sending the same file name to the Linux machine via TCP/IP.

Step 3: The Linux program, which is continuously listening to a certain port now get the name of the file via sockets and does the following processing:

i. If the program executes for the first time, it automatically mounts the remote server directories on its local file system using *mount* command. It also clears up any stale Input text file in the Input directory.

ii. It has to be mentioned here that the Input text file names that are created by the windows program are unique conforming to the following pattern *"day-month-year-hour-min-sec-millisec_Input.txt"*

iii. There is a parser module in the Linux Adapter that reads the file name until _ is encountered and creates a new file with the extension *_Output.txt* and with the same contents in the Output directory.

iv. It copies a sample image file stored in the local file system and renames it with the above patterned *_Output.jpg* and copies it in the Output directory. (This is what that needs to be changed in the final deployment phase.

v. Now, a copy of the image file with the same name is sent to the Moved directory. In addition, the text file in the Input directory is now moved to the Moved directory of the remote computer using Linux's *mv* command. Hence, the Input directory becomes clean with no Input files.

vi. The adapter program again goes back to the listening state.

vii. The adapter then generates the log file. This log files gives the date and time as to when the Input file is created in the Input directory and when the user views the result on browser.

Detailed Flow Diagram

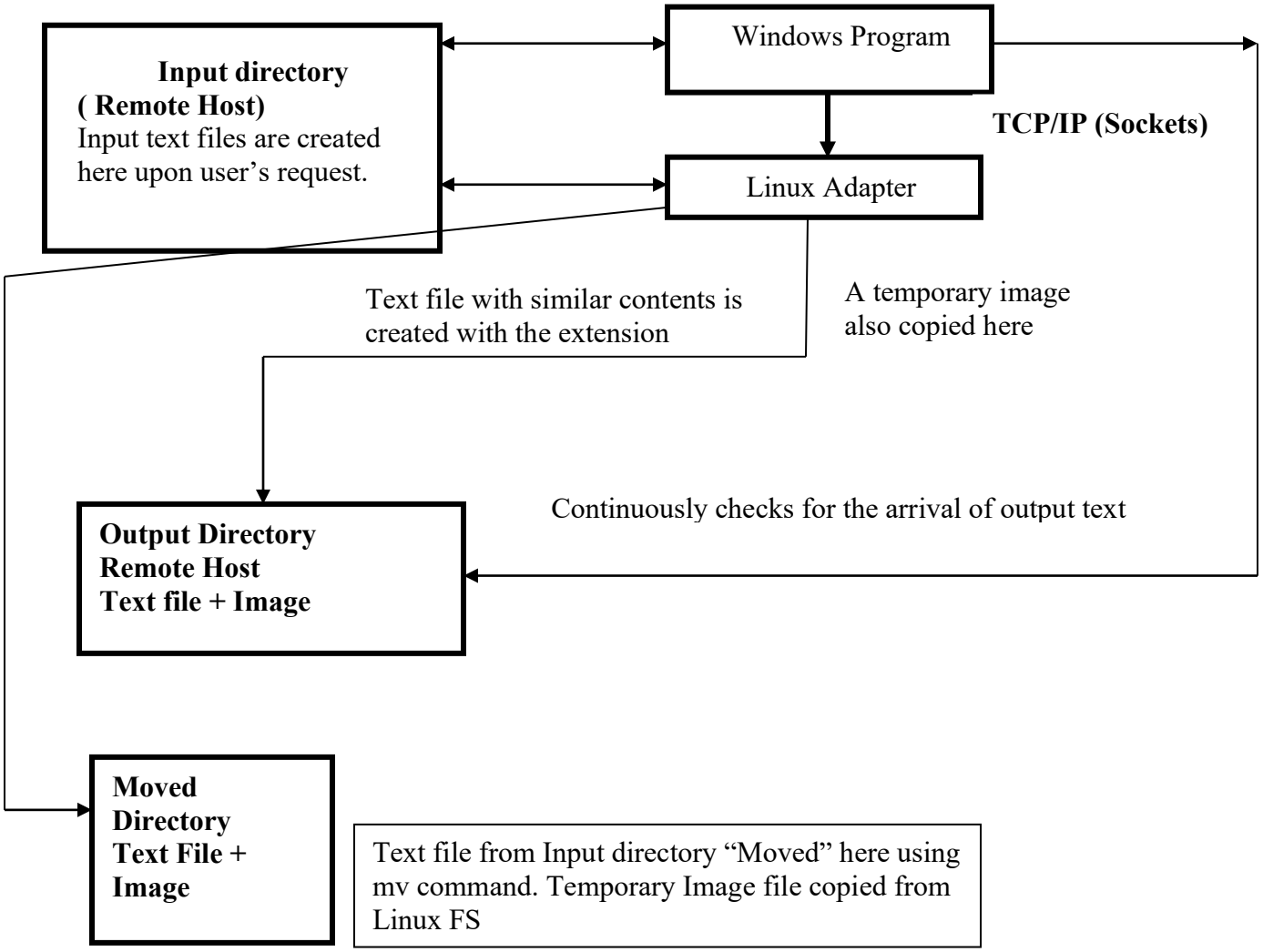
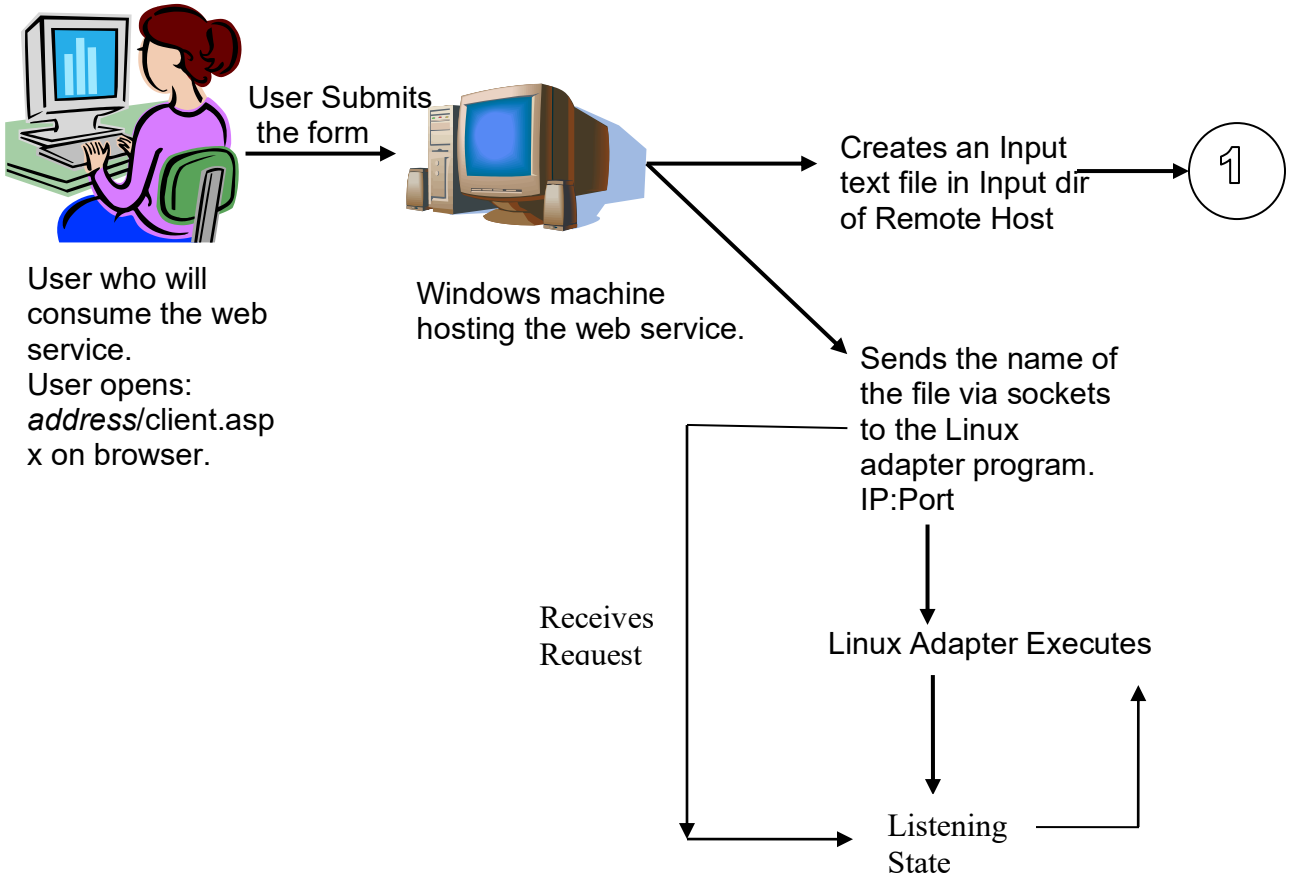


Figure 10: Control Flow



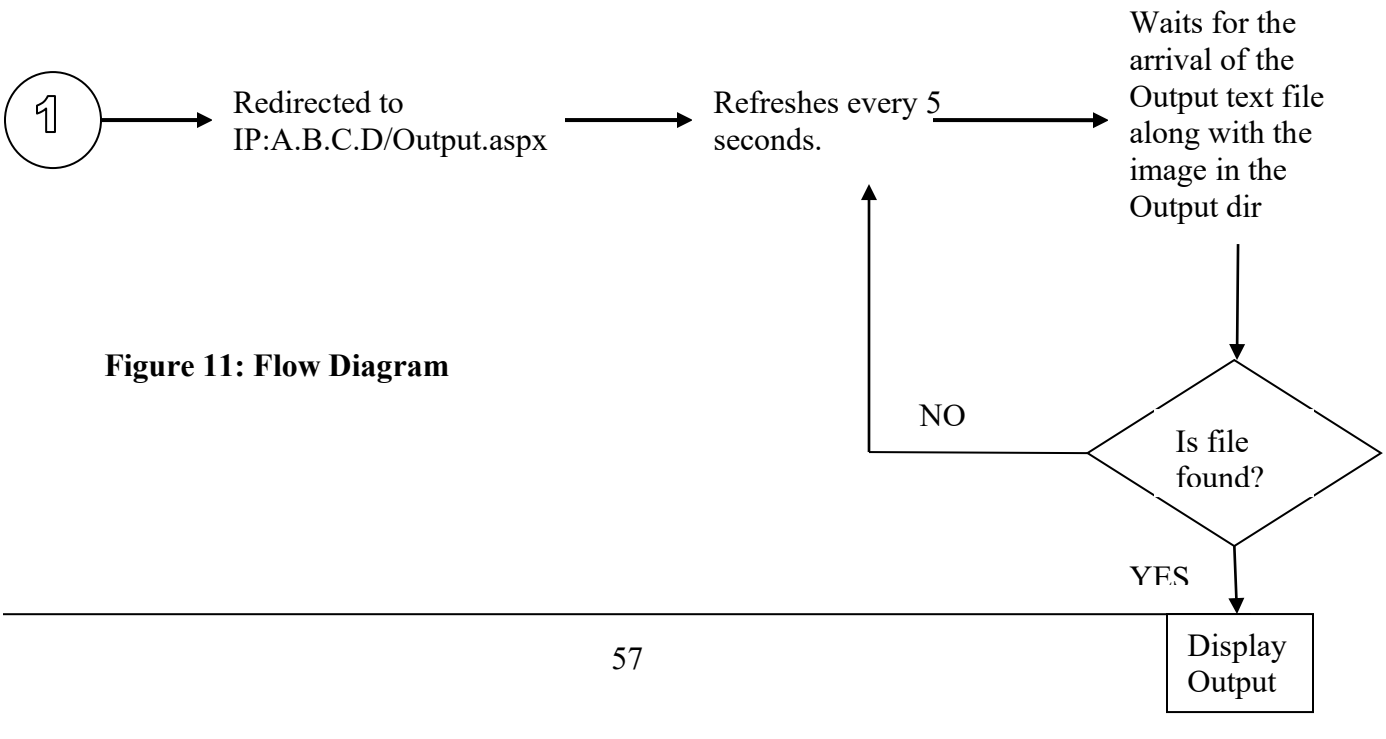


Figure 11: Flow Diagram

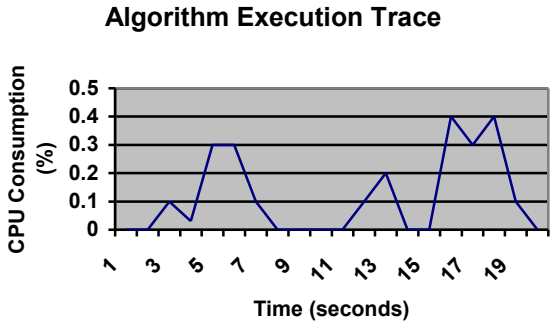
Section 3.4.2.1 ► Why this implementation failed?

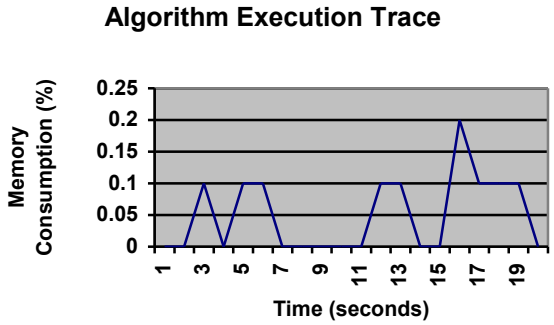
File Input/Output are time consuming and CPU intensive operations. A single disk read consists of a series of CPU and hard disk controller operations. A better and more optimized solution could be mapped on using sockets with no Disk or file I/O.

Communication via TCP/IP is both fast and reliable. For this, sockets need to be implemented at both the ends: GNU/Linux and Windows Web services. The sockets should use one port and should conform the client-server model.

The above implementation involved intensive File I/O, which made the entire flow highly unreliable, CPU and Disk read/write intensive. In order to search for better and more optimized implementation this model was discarded.

Technical Details:





Plot 3: CPU Consumption v/s Time

Plot 4: Memory Consumption v/s Time

Section 3.4.2.2 ► Problems faced and lessons learned

Initially, there was a problem connecting the Linux machine to the windows hosted sockets over TCP/IP. Firewall, antivirus and other security softwares were disabled but the problem still persisted. Later it was noticed that due to some programming flaw at both the ends the two sockets failed to connect using a common port.

A firewall or antivirus application does not normally interfere in socket communications.

TCP/IP is an open standard for communication among open systems. It does not matter whether an application is written in C or Visual Basic, if both the applications use a common port for TCP/IP communication; they are bound to exchange data. This was proved when windows based web service running on Virtual Machine Model communicated with a C application over a TCP/IP socket, which is a low level language.

Section 3.4.2.3 ► Implementation Details ► Pseudo Code

Step 1 START

Step 2 Dynamically initialize all the variables

Step 3 OPEN the file pointers

Step 4 Initialize the socket

Step 5 LISTEN to connections

Step 6 LOOP forever

Step 7 ACCEPT connections

Step 8 RECEIVE data from remote Windows web service. Sends the file name

Step 9 STORE the name of the file in another CHAR array

Step 10 Parse the file name AND extract the unique time stamp

Step 11 Create an image file of the same time stamp with the extension of .jpg in the Output directory

Step 12 Copy the same image file in the Moved directory

Step 13 FREE all the dynamically allocated arrays

Step 14 Go back to ACCEPT state

Section 3.4.3 ► Solution Strategies ► Strategy 3 ► Adapter Implementation Using sockets (TCP/IP) without any Disk I/O

The Computing Setup

1. Operating System: Fedora Core 4
2. Linux Kernel Version: 2.6.11-1.1369-FC-4
3. RAM: 256 MB
4. HDD: 80 GB
5. Processor: Intel Pentium 4
6. Processor Cycle Speed: 2.9 GHz.
7. Programming Language: C
8. Compiler – GNU's gcc ver – 4.0.0

Process Runtime Statistics

ps –aux | grep *file_name.out* gave

CPU Usage = 0.0% when in listening mode
Memory Usage = 0.0% (in listening mode)

When the Adapter executed

top command showed:

CPU (%) = 0.1
Memory (%) = 0.1

Section 3.4.3.1 ► Base 64 Encoding

The Base 64 encoding is designed to represent arbitrary sequences of octets in a form that requires case sensitivity but need not be humanly readable. A 65-character subset of US-ASCII is used, enabling 6 bits to be represented per printable character. (The extra 65th character, "=",is used to signify a special processing function.) The encoding process represents 24-bit groups of input bits as output strings of four encoded characters. Proceeding from left to right, a 24-bit input group is formed by concatenating 3 8-bit input groups. These 24 bits are then treated as 4 concatenated 6-bit groups, each of which is translated into a single digit in the base 64 alphabet. Each 6-bit group is used as an index into an array of 64 printable characters. The character referenced by the index is placed in the output string.

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

Table 1: The Base 64 Alphabet

Special processing is performed if fewer than 24 bits are available at the end of the data being encoded. A full encoding quantum is always completed at the end of a quantity. When fewer than 24 input bits are available in an input group, zero bits are added (on the right) to form an integral number of 6-bit groups. Padding at the end of the data is performed using the '=' character. Since all base 64 input is an integral number of octets, only the following cases can arise:

1. The final quantum of encoding input is an integral multiple of 24 bits; here, the final unit of encoded output will be an integral multiple of 4 characters with no "=" padding,

2. The final quantum of encoding input is exactly 8 bits; here, the final unit of encoded output will be two characters followed by two "=" padding characters, or

3. The final quantum of encoding input is exactly 16 bits; here, the final unit of encoded output will be three characters followed by one"=" padding character.

Flow Diagram

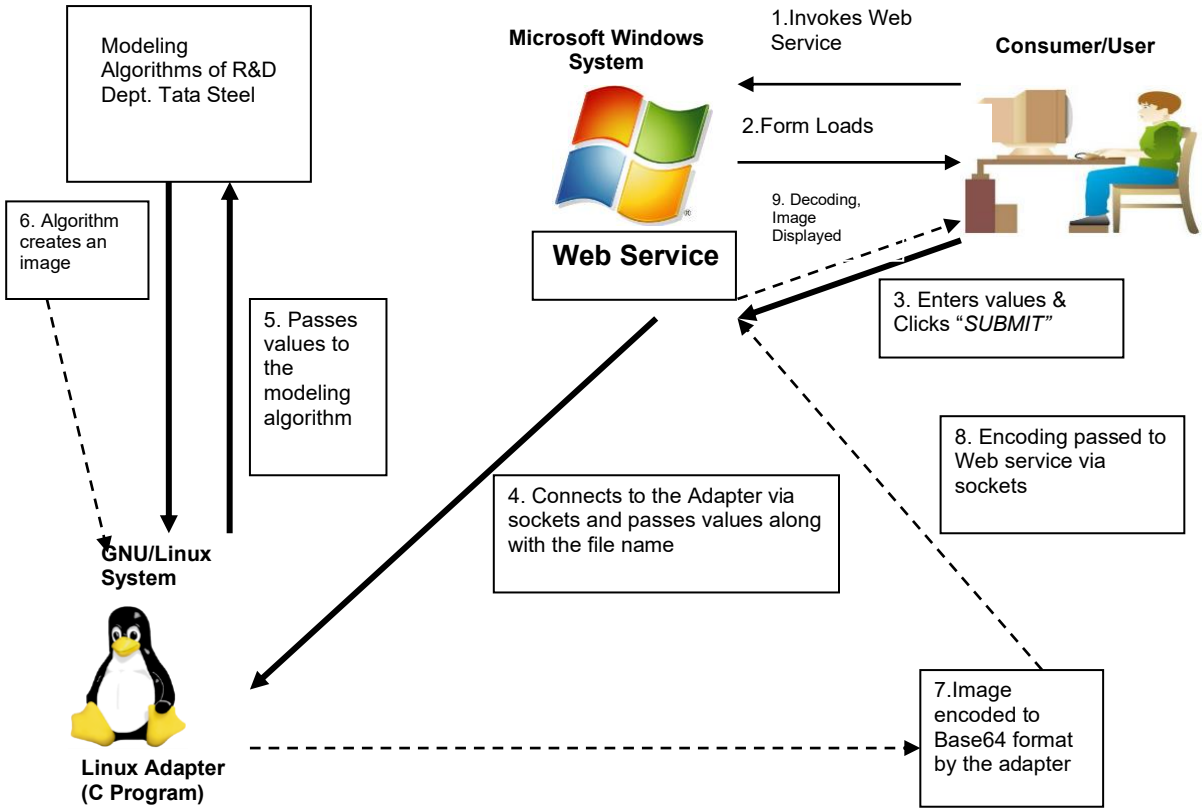


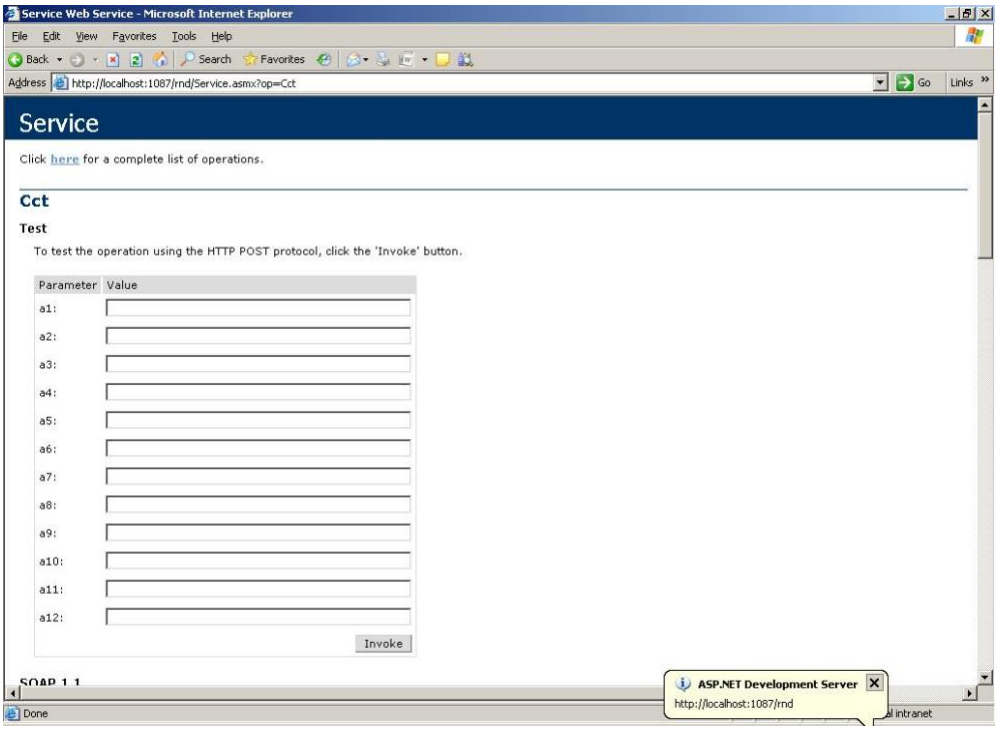
Figure 12: Linux Adapter using Sockets (TCP/IP): Architecture

Explanation

This implementation used sockets with no Disk Read/Write access. This is what makes it a complete generic web service implementation.

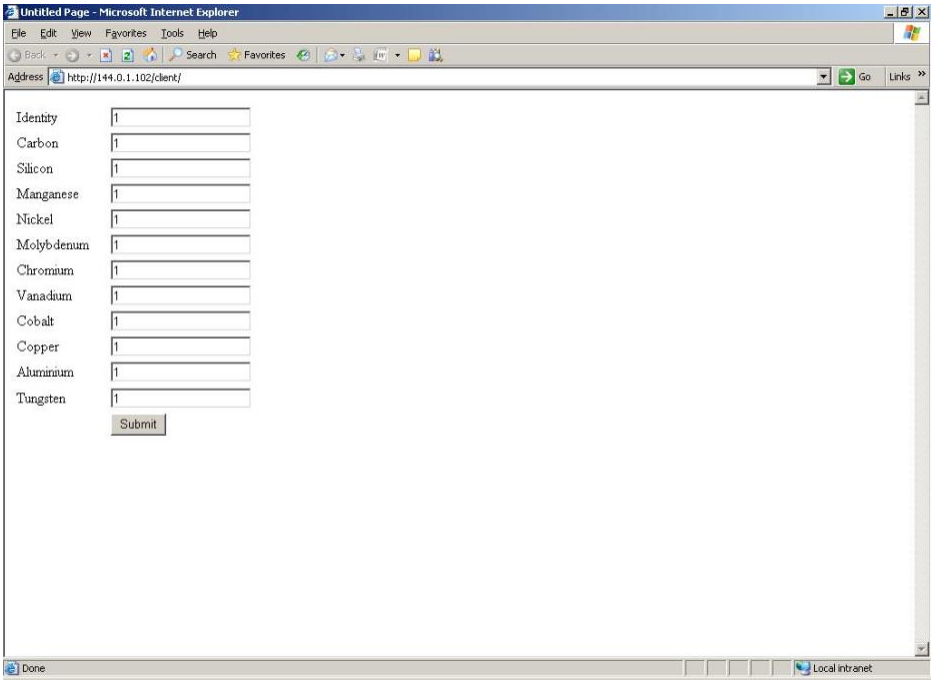
The complete architecture is explained below:

1. There is a server, which hosts the web service. This is the server from where all users download the web service's URL onto their browsers. This web server is a Microsoft Windows machine running Windows Operating System.



Screen 4: Web Service

2. Let us bring a user in action. This user downloads the URL onto his/her browser. This is referred to as invoking the web service at the server's end. The server now sends the requested web page to the user.



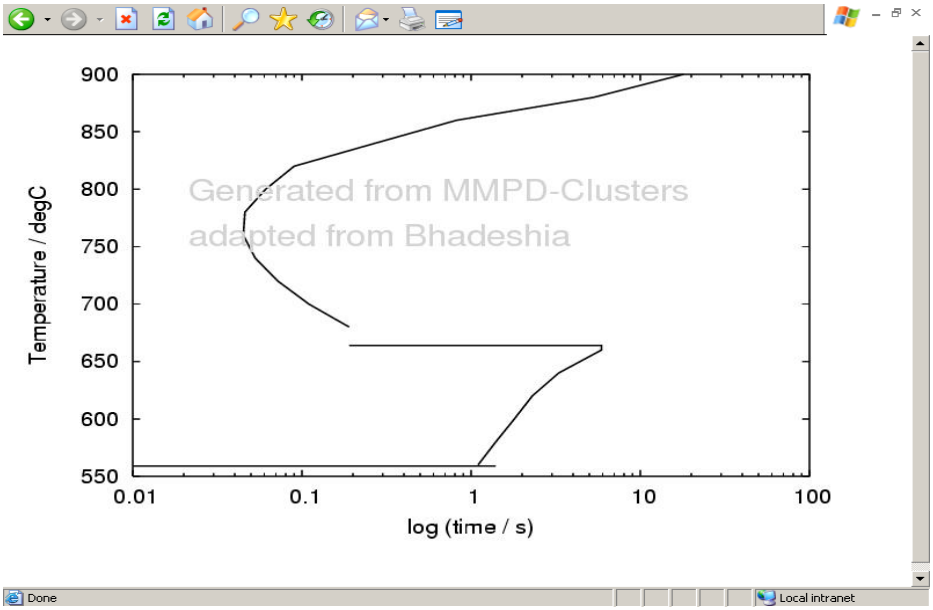
Screen 5: The form that the user sees at his/her end

3. The web page that the user sees is a form, which requires the user to fill in with some numerical values.
4. The user fills in the form and clicks **Submit** button.
5. The web service gets the user's values and stores it in an array.
6. The web service now connects to another program, running on the Linux machine. This is a C program, which uses BSD Socket API. This is C Program is the Linux Adapter.
7. The Web service connects to this adapter via TCP/IP sockets.
8. The web service now sends the entire array to the Linux based adapter to do the rest of the processing. The array is of the format **number#number#number#number%time_stamp**
9. Linux adapter in turn sends the values to the modeling algorithms deployed at R&D Department for modeling calculations using the values that the user has entered. These algorithms run on a Linux Cluster.
10. The modeling algorithm, after processing the values, generates an image, which is in JPEG format.

11. This image is dumped onto a shared directory, which the Linux adapter can use.
12. The adapter, the moment it gets the image, converts it into Base64 encoding and stores the encoded data in an array, subsequently sends it to the Windows hosted web service.

Screen 6: Base64 Encoding received by the web service from Linux Adapter

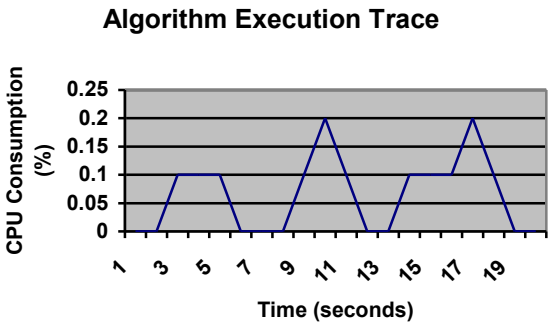
13. The web service decodes the data and displays it as an image.



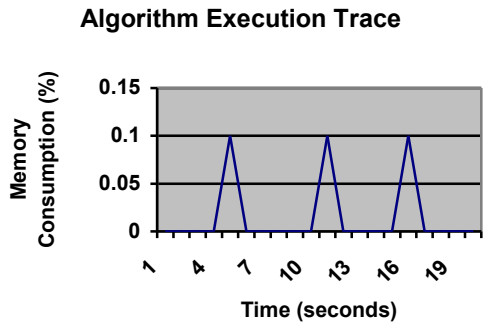
Screen 7: Image Displayed

Section 3.4.3.2 ► Reasons for its success

1. The implementation of this Linux Adapter is highly optimized. This implementation consumed 0.0% of the entire CPU and 0.0% of Memory during the No Operation state.
2. When the adapter executed, it consumed about 0.2% of the entire CPU value and 0.1% of the entire memory. Hence, the best possible solution
3. Communication was done using TCP/IP protocol. TCP/IP is both reliable and fast.
4. Disk I/O was done away with.
5. **Technical Details:**



Plot 5: CPU Consumption v/s Time



Plot 6: Memory Consumption v/s Time

Section 3.4.3.3 ► Problems faced

1. This implementation required lot of labour, which tested the competencies of everyone involved in this project to oceanic extent. The major problem was related to sending of base64-encoded data to the windows machine. These were the observations that were made:
    - When Windows based web service sent any number of characters to the Windows based socket, the socket consumed all the characters in one read operation at the receiving end.
    - Linux to Linux over TCP/IP socket mounted no problems.
    - The problem occurred when the Linux adapter sent the encoded data to the Windows based system; the Windows system accepted some data and rejected the

rest. This happened in cases where the size of the image was large and hence more number of characters in the encoded base64 data. The Linux adapter's *send()* function returned the total number of bytes sent. This corresponds to the number of characters in the encoded data. However, the Windows machine did not receive the entire byte stream. This, until now remains a mystery and no reason has been discovered as to why this acute problem persists.

2. Decoding of the encoded data, at Windows end, sent by the Linux adapter posed another serious problem. Linux socket sends data as string. The web service was able to consume that string data but was unable to decode it. This problem was later solved, which existed at Windows end.
3. C programs due to their inherent design execute very fast. C/C++ programs are far more efficient as far as execution time goes. None of the modern programming languages can match the execution speed of the C/C++ programs. What was seen during the implementation was that as the web service sent the data to the adapter, which was written in C. The adapter executed so fast and completed its operation before the web service was ready with its data receive via socket. The problem was resolved by explicitly reducing the execution speed of the C program using *sleep()*

Section 3.4.3.4 ► Lessons Learnt

1. Sockets programming is all about synchronization of *send()* and *recv()* at both ends between a client and the server.
2. It is not just image that can be transported to the remote host using base64 encoding but other file formats like *.doc, .xls, .avi* i.e. MIME types, can also be transferred from a Linux socket to the web service.
3. Base64 encoding facilitates faster and more reliable sending of files over the network in encoded format.

Section 3.4.3.5 ► Implementation Details ► Pseudo Code

Step 1 START

Step 2 Dynamically initialize all the variables

Step 3 OPEN the file pointers

Step 4 Initialize the socket

Step 5 LISTEN to connections

Step 6 LOOP forever

Step 7 ACCEPT connections

Step 8 RECEIVE data from remote Windows web service. Sends the values along with the unique time stamp value. The format is **number#number#number#number#number%time_stamp**

Step 9 STORE the name of the file and values in separate CHAR arrays

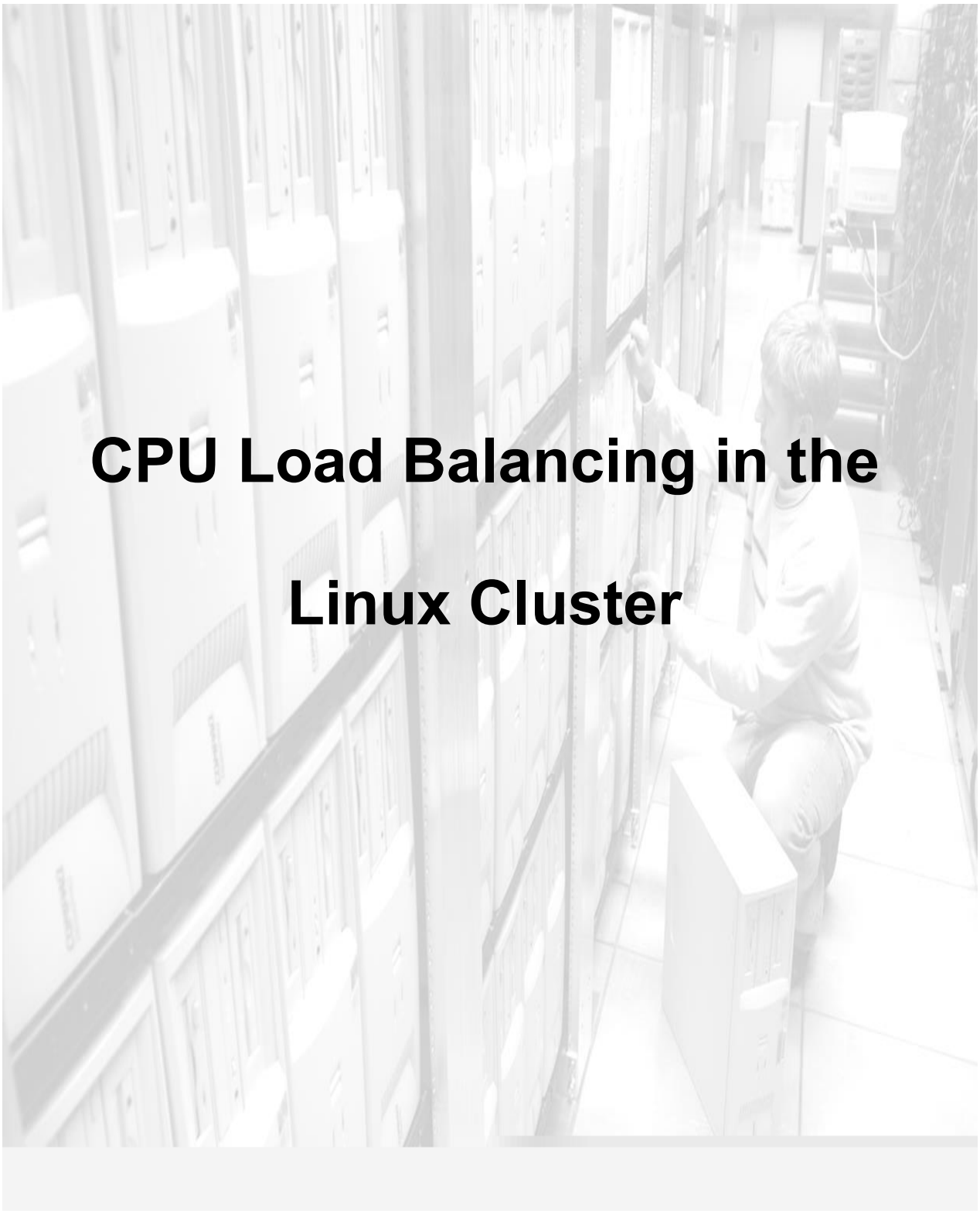
Step 10 STORE the values in a file; to be processed by modeling algorithm

Step 11 Get the image from local drive and convert to BASE64 Encoding

Step 12 STORE this encoding in a CHAR array and WRITE to Windows socket

Step 13 Windows socket does the rest

Step 14 Go back to ACCEPT state



CPU Load Balancing in the Linux Cluster

Section 4.1 ► Introduction

Section 4.1.1 ► Computing Cluster

A computer cluster is a group of coupled computers that work together closely so that in many respects they can be viewed as though they are a single computer. The components of a cluster are commonly, but not always, connected to each other through fast local area networks. Clusters are usually deployed to improve performance and/or availability over that provided by a single computer, while typically being much more cost-effective than single computers of comparable speed or availability.

Section 4.1.2 ► Cluster categorizations

- **High-availability (HA) clusters**

High-availability clusters (also known as failover clusters) are implemented primarily for the purpose of improving the availability of services which the cluster provides. They operate by having redundant nodes, which are then used to provide service when system components fail. The most common size for an HA cluster is two nodes, which is the minimum requirement to provide redundancy. HA cluster implementations attempt to manage the redundancy inherent in a cluster to eliminate single points of failure.

There are many commercial implementations of High-Availability clusters for many operating systems. The Linux-HA project is one commonly used free software HA package for the Linux OSs.

- **Load-balancing clusters**

Load-balancing clusters operate by having all workload come through one or more load-balancing front ends, which then distribute it to a collection of back end Platform LSF HPC, Sun

Grid Engine, Moab Cluster Suite and Maui Cluster Scheduler. The Linux Virtual Server project provides one commonly used free software package for the Linux OS.

Section 4.1.3 ► Technologies

MPI is a widely-available communications library that enables parallel programs to be written in C, Fortran, Python, OCaml, and many other programming languages.

The GNU/Linux world supports various cluster software; for application clustering, there is Beowulf, distcc, and MPICH. Linux Virtual Server, Linux-HA - director-based clusters that allow incoming requests for services to be distributed across multiple cluster nodes. MOSIX, openMosix, Kerrighed, OpenSSI are full-blown clusters integrated into the kernel that provide for automatic process migration among homogeneous nodes. OpenSSI, openMosix and Kerrighed are single-system image implementations.

Microsoft Windows Compute Cluster Server 2003 based on the Windows Server platform provides pieces for High Performance Computing like the Job Scheduler, MSMPI library and management tools. NCSA's recently installed Lincoln is a cluster of 450 Dell PowerEdge 1855 blade servers running Windows Compute Cluster Server 2003. This cluster debuted at #130 on the Top500 list in June 2006.

DragonFly BSD, a recent fork of FreeBSD 4.8, is being redesigned at its core to enable native clustering capabilities. It also aims to achieve single-system image capabilities.

Section 4.2 ► The Scenario at R&D Department

The R&D Department of Tata Steel has purchased a cluster consisting of seven or more computers. This cluster would consist of CPUs running Linux based operating systems. The challenge is that of CPU load balancing and process migration. Apart from modeling algorithms, other CPU intensive jobs will also be carried using this Linux cluster.

Now the design requires that if one of the CPUs is busy with some calculations, the other CPU should take over with the new set of calculations that arrive at the cluster.

Section 4.3 ► Computing Setup

The Computing Setup

1. Operating System: Fedora Core 4
2. Linux Kernel Version: 2.6.11-1.1369-FC-4
3. RAM: 256 MB
4. HDD: 80 GB
5. Processor: Intel Pentium 4
6. Processor Cycle Speed: 2.9 GHz.
7. Programming Language: C
8. Compiler – GNU's gcc ver – 4.0.0
9. Number of CPUs during initial development: 2
10. Total number of CPUs supported by this architecture: 255

Section 4.4 ► Design Phase

The following design gives an idea of cluster computing. **It is important to mention that the scope of this design is limited to just the adapter and modeling calculations. It does not take care of the rest of process migration stuff.**

Here is the proposed architecture of the cluster.

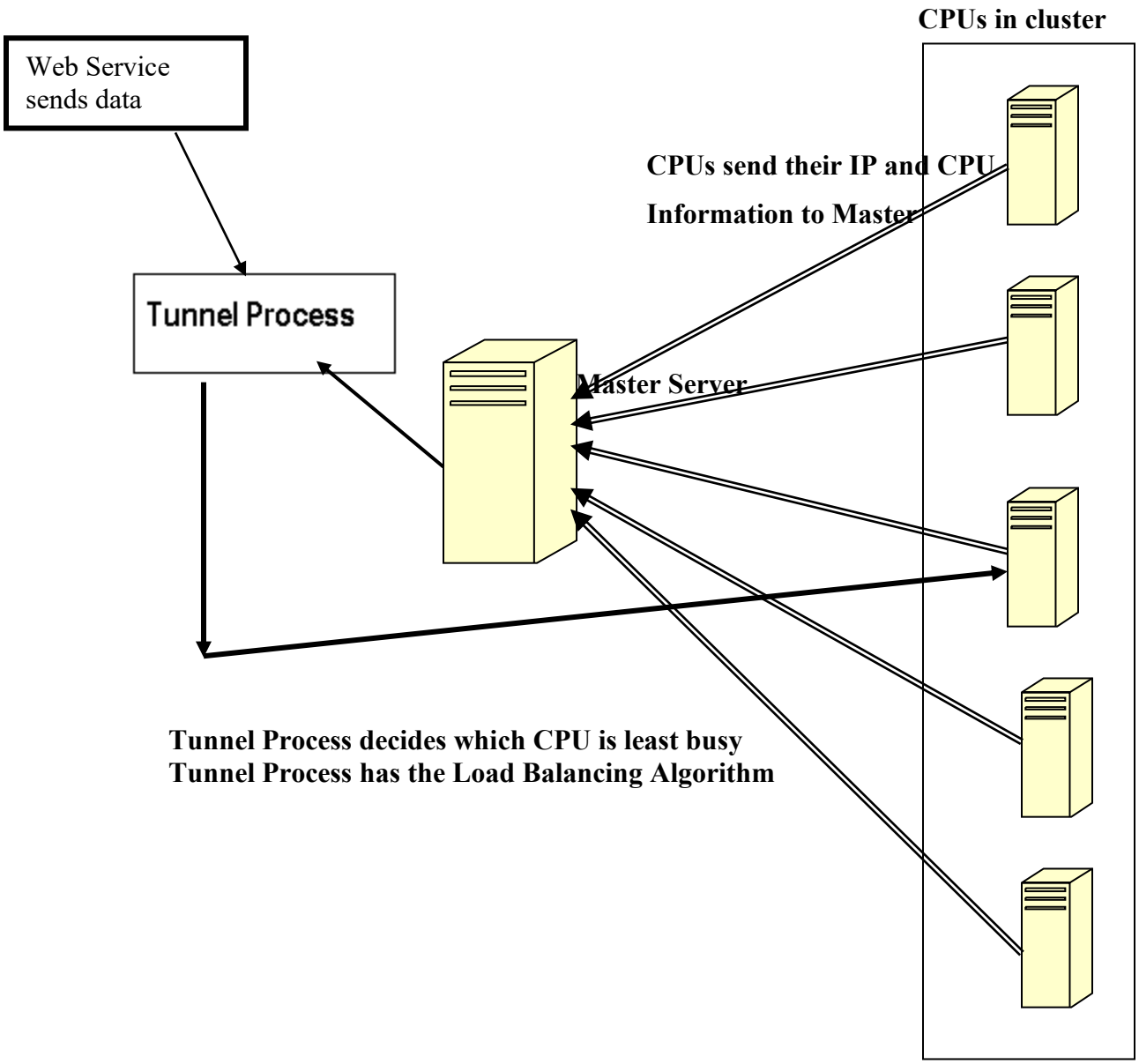


Figure 13: The Design of CPU Load Balancing in Linux Cluster

Section 4.4.1 ► Assumptions

1. Master server is always up and running. If the master server goes down the entire architecture fails.
2. It does not matter if any machine in the cluster other than the master server crashes.
3. If all the machines in the cluster are down, the master server takes over the modeling calculations.
4. The Linux Adapter need to be up and running on all the machines in the cluster and all adapters must use a common port.
5. The client programs should be installed on every machine in the cluster except the master server.
6. The client programs must use one port and their connecting IP must be the IP of the master server.
7. The master server is not some special computer but it is one among the computers in the computing cluster that handles an extra task of maintaining the central database.
8. There is another program called the tunnel program or the Load Balancing Program that receives the values from the web service and the database values from the master server.
9. The Load Balancing Algorithm decides which CPU in the computing cluster has the least load.
10. This is not a general-purpose load-balancing algorithm. This entire architecture is catered to the requirements of the modeling computations and is hooked to the Linux Adapter.
11. This architecture will run on any Linux, UNIX and Solaris based systems.

Section 4.4.2 ► Explanation

1. One computer from the cluster is chosen as the Master Server.
2. The role of the master server is to maintain the database consisting of IP and CPU information.
3. Every machine in the computing cluster connects to the master server every one minute and sends its CPU and IP information.

4. Communication is facilitated via sockets.
5. The master server, as it completes the database, sends the data base information to the Tunnel Process.
6. The tunnel process has in built searching and sorting algorithms that decide which CPU is free and where the task has to be uploaded.
7. Web service also connects to the same tunnel program.
8. The Tunnel Process has a data check algorithm, which checks from which source data is received and hence does the processing accordingly.
9. The Master Server is just like other computers in the cluster. It is entrusted with the special task of maintaining the central database.
10. CPU usage indicates the CPU consumption by the processes. Expressed as percentage.
11. The design of the central database is as given below:

IP	CPU
120.44.83.21	**89.7**
10.9.8.7	**10.43**
1.2.3.4	**0.004**
5.4.3.2	**1.30**
133.0.0.43	**90.08**

Table 2: Central Database

Section 4.5 ► Implementation Phase

Section 4.5.1 ► The Master Server

This is one special machine chosen among the machines of the computing cluster that does the task of maintaining the CPU and IP information from all other machines in the cluster. Every machine in the cluster connects to the master server after 1 minute and sends it their CPU and IP information. The communication is facilitated via sockets. The entire architecture has been implemented in C programming language due to efficiency reasons.

Section 4.5.2 ► Pseudo code

Step 1 START
Step 2 OPEN the database file for writing
Step 3 INITIALIZE two IP addresses
Step 4 INITIALIZE two Ports
Step 5 INITIALIZE socket for connection one
Step 6 LISTEN for connections
Step 7 INITIALIZE socket two for another connection
Step 8 ACCEPT connections
Step 9 REPEAT UNTIL all the machines in the cluster have sent in their CPU information
Step 10 GET CPU status information from itself
Step 11 SUM the total CPU usage
Step 12 STORE all the values in a text file
Step 13 CLOSE the file
Step 14 OPEN the file and SEND the contents of the file via socket to load balancer
Step 15 DELETE the database file
Step 16 FREE all the dynamically allocated arrays
Step 17 Go back to LISTEN state

Section 4.5.3 ► Reliability Calculations

Below are some mathematical foundations related to this supercomputing cluster.

Probability of task completion (task relates to the set of values that the web service sends to the executable) in circumstances when the master server is down = 1

This means that if the master server is down, the entire modeling operation will not fail.

> **Assumption: Load Balancer and Master Server programs are on different computers in the computing cluster.**

Proof:

Let *M* denote the Master Server.

Let *C1, C2, C3…*etc denote the other machines in the computing cluster.

Task is bound to complete if the load balancer or tunnel is up and running.

Therefore, Probability that the task is complete when the master server (M) is up = 1.

Reason being, task is bound to complete if any one machine in the computing cluster is up and running.

Reliability is increased when the load balancer and the master server programs are on different computers in the computing cluster.

The load balancer has a default connection IP in case of failures. This IP is the address of the machine itself from where the load balancer is executing.

Probability of task completion when the Load Balancer is down = 0

It does not matter whether the Master Program and the tunnel are on same or different computers in the computing cluster. Since, the web service connects to this tunnel algorithm, when this is down web service will fail to connect.

Probability of task completion when the master server is up and running (tunnel algorithm is on a different computer in the cluster, which is up and running) = 1

This can be proved from the results derived above.

Therefore, the probability, that the task is complete when the load balancer is up and any other CPU (C1,C2 or C3 *etc*) in the cluster is down = 1

Proof:

Task will complete if any other machine in the cluster is down as the load balancer is up. From the results proved above, the probability of task completion is 1 when the Load Balancer is up and running.

Probability of completion of task when the master server and the load balancer is on the same machine and the machine is down = 0

The entire architecture fails, in this case.

Section 4.5.4 ► Conclusion

This design is highly reliable and fault tolerant. Problem occurs only when the load balancer is down; the entire system comes to a halt. This problem can be resolved only when the web service is able to connect to different machines after sensing that the load balancer is down. This task has to be done at the Windows end. The best bet is to run the tunnel and the master server cluster programs on different computers in the computing cluster. Reason being, even if one of them is down the system will keep functioning without fault. The load balancer will take over the modeling calculations in such a case.

Anyway, this scheme is highly reliable and fault tolerant.

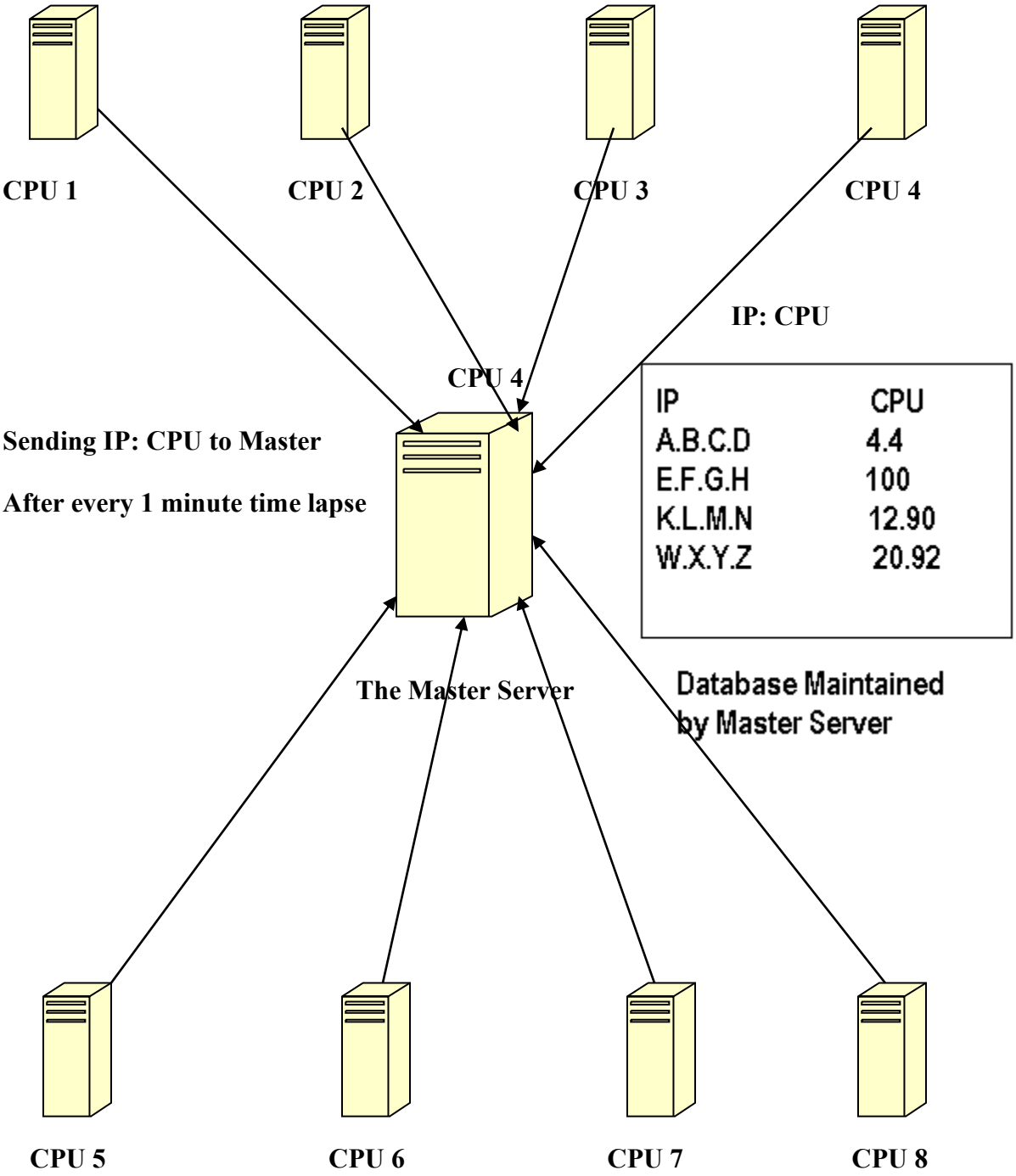


Figure 14: The Master Server

Section 4.5.5 ► The Load Balancing Algorithm or Tunnel Algorithm

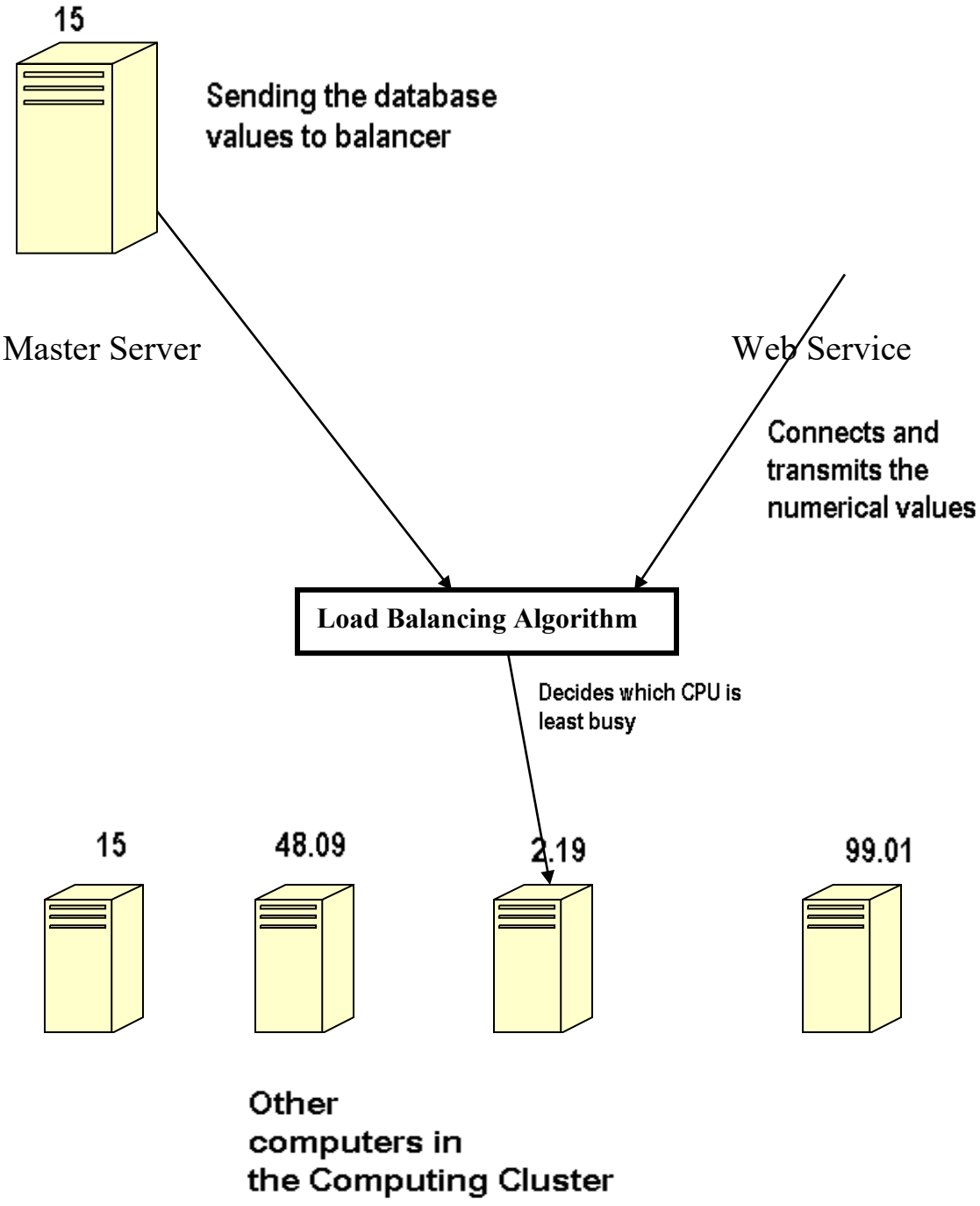


Figure 15: Load Balancer

Section 4.5.6 ► The Tunnel or the Load Balancer

The load balancer does the task of searching and assigning the CPU with the minimum usage. The load balancer connects to two ports via sockets to two different applications one the master server and the other is the web service. It has an in-built data check as to from which source the data has come. If the data is from the web service, the balancer checks the CPU values and calculates which CPU has the least usage and forwards the values to the corresponding adapter. If data is from the master server, the load balancer recalculates the CPU usage and keeps it handy. This process of receipt of data from the master server repeats every 1 minute. This ensures that the data is fresh.

The master server has two variables that are static in nature. One is the destination IP of the CPU, which is minimal in use and the second the CPU value, which is least.

Section 4.5.7 ► Pseudo Code

Step 1 START
Step 2 INITIALIZE the IP
Step 3 INITIALIZE the Port
Step 4 INITIALIZE the array for central database values receipt
Step 5 INITIALIZE the socket
Step 6 LISTEN to port
Step 7 ACCEPT connections
Step 8 RECEIVE the values if connection TRUE
Step 9 IF values from web service
Step 10 FIND the IP to connect to adapter from the IP_STORE variable
Step 11 IF IP_STORE variable is NULL
Step 12 THEN master server processes the modeling
Step 13 ELSE send the values to the minimum CPU value
Step 14 GOTO Step 21
Step 15 IF value is from master

Step 16 THEN process and STORE the IP and CPU in respective structures

Step 17 FIND the minimum CPU value

Step 18 FIND the corresponding IP value

Step 19 STORE the IP in IP_STORE variable

Step 20 FREE all dynamically allocated variables except IP_STORE

Step 21 Go back to LISTEN state

Section 4.5.8 ► Problems Encountered

Designing applications for fault tolerant systems is a mammoth task. It not only tests one's programming abilities but also touches every level of the designer's competence. Be it operating systems or network programming concepts.

1. The first and foremost problem pertains to the design of this architecture itself. Several person-hours were spent designing the architecture. One not only has to understand the circumstances and the hardware setup under which the system would function but also every operation has to be done in an optimized fashion, so that the architecture does not consume large amounts of CPU.
2. **Optimization related Problems:** Coding was done on C. Therefore, there was an advantage of efficiency and speed of execution. However, one has to apply one's own knack to draw the final code that is both optimized and does not bother the CPU much. Code related to searching and master server data base maintenance has to be optimized. In order to optimize code, the client machines were handed with the task of calculating their own CPU and getting their IP addresses. Better, optimization was added when 0 were removed from the CPU summation calculations. Therefore, the task of addition module was to add non-zero numbers.
3. **Readers-Writes Problem** [20]**:** In computer science, the readers-writers problem is an example of a common computing problem in concurrency. The problem deals with situations in which many threads must access the same shared memory at one time, some reading and some writing, with the natural constraint that no process may access the share for reading or writing while another process is in the act of writing to it. (In

particular, it is allowed for two readers to access the share at the same time.) A readers-writer lock is a data structure that solves one or more of the readers-writers problems.
The initial design used to make use of a single file, which is the central database. The master server used to do its task of writing to the database simultaneously the tunnel or the load balancer used to read the same database. This design completely suffered a serious setback when it was found that the load balancer could not retrieve any values from the database because the master server is updating it. Hence, the entire design failed.

This problem cannot be solved by file locking mechanism owing the fact that if the number of computers in the cluster is large then most of the time the master server will keep the file locked for writing. This will delay the modeling calculations. The challenge was to do tasks in real time. It was the top priority to reduce the waiting time of the user.

Therefore, file locking mechanism was not the right solution.

Finally, the problem was solved using sockets. It was noticed through the source codes that there is an instant in the master server's execution, when it flushes all the data to the central database and closes the file for a moment. Now, at this instant, the file is fully updated and any application can read it. The entire file contents were read and stored in an array, which was dynamically allocated using *calloc()*. The array contents were passed to another application through sockets. This ensured that the array contents that has to be finally read during allocation for minimum CPU is not from a file but dynamically taken from an array. This keeps the Readers-Writes Problem at bay. The result is that whenever the load balancer wants the array for minimum CPU calculations it will always get the complete information of CPU and IP of all machines in the cluster.

If a situation arises when the web service and the master server send the values to the load balancer simultaneously, then one of them has to be queued for a while. This is what can be referred to as the disadvantage here. However, lot of optimization has been incorporated in the tunnel algorithm, which makes it execute at blistering speeds. It has been tested that the user does not have to wait a long time (less than 1 second) in order to get his/her turn. This is the instance where the efficiency of C binaries comes into picture.

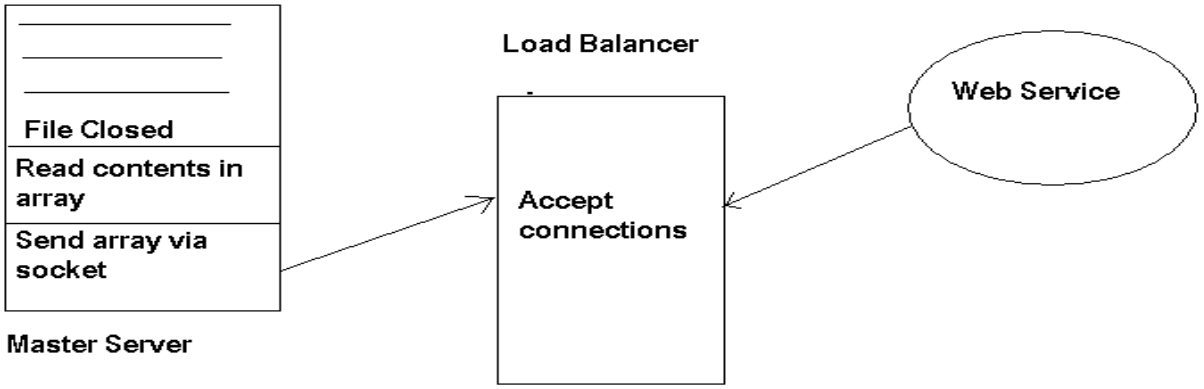


Figure 16: Solving Readers-Writers Problem using sockets

Section 4.5.9 ► Lessons Learnt

1. Discovered a unique way of solving the readers-writers problem. However, this solution is not applicable to every instance where similar kind of problems exists like Operating Systems. This kind of solution is situation dependant.
2. Connecting more than one socket through a single application is indeed possible.

Section 5 ► Summary and Conclusion

Section 5.1 ► Summary of Achievements

1. Successfully designed and deployed the Apache Web server.
2. Successfully designed, coded and deployed the Linux adapter. This adapter is currently executing on the R&D server and serving requests from Corus.
3. Designed, coded and deployed the CPU Load Balancing architecture.

Section 5.2 ► Some Special Observations

1. Two different families of sockets; one running on Windows using *winsock* and other UNIX sockets adhering to BSD socket API, were able to communicate and send, receive large bytes of data.
2. A programming language very close to the system i.e. C in Linux, was found to be fully compatible with the three tiered based applications running on virtual machines i.e. web services, as far as the communication through socket is concerned.
3. Reasons as to why the Windows machine discarded large amounts of data when the Linux machine sent to it via sockets is still a mystery. The problem was not encountered when the number of bytes was small. However, if number of bytes was very large ranging in millions or billion number of characters, the Windows machine consumed some and discarded the rest.

Section 5.3 ► Future Challenges

1. This web service will be made available to Tata Steel's subsidiary companies in Malaysia, China and Singapore.
2. Providing Multi-lingual support. Current implementation supports only English.
3. Security is another concern as information is exchanged through open ports.
4. XML poisoning has to be dealt with.

Section 5.4 ► Final Word

It was a great ride covering programming challenges to deployment brainstorming. The ride was also characterized by getting to know and work with scholars like Dr. M. Murugananth. This journey was an atmosphere studded with challenges, exploration and knowledge sharing. Working with scholars like Dr. Murugananth was an opportunity of a lifetime. I must admit that this is the first time I have met any personality of such high caliber, academic excellence, qualifications and international recognition. Getting industrial exposure on such a massive scale

was also something new that I have attained during my internship tenure at Tata Steel Limited. The best part was; I had full liberty to research and apply my own creativity in the problem-solving scenario. Interacting with engineers from Microsoft was a rare occasion that I got. I thank my project guide for this wonderful opportunity. After all, I must admit that Microsoft is a great place to work.

Section 6 ► References

[1] Tata Steel Limited Website [Online]: Available: *http://www.tatasteel.com*

[2] Tata Steel [Online]: Available: *http//www.en.wikipedia.org/wiki/Tata_Steel*

[3] Jamsetji Tata [Online]: Available: *http://www.en.wikipedia.org/wiki/Jamshedji_Tata*

[4] Dorabji Tata [Online]: Available: *http://www.en.wikipedia.org/wiki/Dorabji_Tata*

[5] J.R.D. Tata [Online]: Available: *http://www.en.wikipedia.org/wiki/J_R_D_Tata*

[6] Ratan Tata [Online]: Available: *http://www.en.wikipedia.org/wiki/Ratan_Tata*

[7] World Steel Dynamics Website [Online]: Available: *http://www.worldsteeldynamics.com*

[8] IBM Mainframes Website [Online]: Available: *http://www.ibmmainframes.com*

[9] Enterprise Resource Planning [Online]: Available:
http://www.en.wikipedia.org/wiki/Enterprise_resource_planning

[10] Apache HTTP Server Documentation [Online] Available:
http://httpd.apache.org/docs/trunk/

[11] Sinha, P.K., *Distributed Operating Systems: Concepts and Design*, *Wiley-IEEE Press* 1996

[12] Distributed Computing [Online] Available:
http://en.wikipedia.org/wiki/Distributed_computing

[13] Corus Website [Online] Available: *http://www.corusgroup.com*

[14] Supercomputer [Online] Available: *http://www.en.wikipedia.org/wiki/Supercomputer*

[15] Load Balancing (computing) [Online] Available:
http://www.en.wikipedia.org/wiki/Load_balancing_(computing)

[16] Hurwitz, J., Bloor, R., Baroudi, C., Kaufman, M., *Service Oriented Architecture for Dummies*, Wiley Publishing Inc. 2007

[17] Web Services Description Language (WSDL) Version 2 Part 2[Online] Available:
http://www.w3.org/TR/2004/WD-wsdl20-patterns-20040326

[18] Simple Object Access Protocol (SOAP) 1.1 [Online] Available:
http://www.w3.org/TR/2000/NOTE-SOAP-20000508

[19] Newcomer, Eric; Lomow, Greg (2005). *Understanding SOA with Web Services. Addison Wesley.* ISBN 0-321-18086-0

[20] Stallings, W., *Operating Systems: Internals and Design Principles*, 4/E, *Pearson Publishers*

[21] Corus Website [Online] Available: http://www.corusgroup.com