NIMBLE@ITCECGRID NOVEL TOOLKIT FOR COMPUTING WEATHER FORECASTING, PI AND FACTORIZATION INTENSIVE PROBLEMS

Er. Vijay Dhir, Assistant Professor, SBBSIET, Padhiana, Distt: Jalandhar
vijaykdhir@yahoo.com

Dr. Rattan K Datta, Advisor(Retd), MET, Govt. of India, Former Professor & Head, SBBSIET, Padhiana, Distt: Jalandhar
rkdatta_in@yahoo.com

Dr. Maitreyee Dutta, Associate Professor, Department of Computer Science & Engineering, N.I.T.T.R, Chandigarh
d_maitreyee@yahoo.co.in

ABSTRACT

The processing power of digital computers has increased by a factor of ten raised to power twelve in the last six decades. The growth in power is following well known Moore’s law. At the same time storage capacity has also increased almost exponentially. RAM of a few thousand kilobytes was a great memory in 1950/60s, today normal desk tops carry a memory of a few Giga bytes & disc storage of tera to peta bytes. The efficiency of software has also kept pace with the growth of H/W. Not only it has become user friendly, speedier but also significantly more efficient.

Despite all the above mentioned developments, the researchers seek more processing power & more efficient algorithms to model & solve their problems.

Several complex models have been developed by scientists and these models are programmed by the power of the computer to solve mathematical and physical universe problems such as weather forecasting, climatic assessment, environmental studies, modeling of nuclear energy etc. More the processing power of the computer, precise & closer could be the result to the actual and less the time it will take. To solve the types of complex problems mentioned above, one needs to carry out hundreds of peta operations. To get usable results in time, the speed of the computer should be in peta FLOPS or even more. We can speed up the computation by using very powerful supercomputers (multiprocessors) working with a large number of processors. This requires an investment of over a few crores of Rupees.

Another approach which became available in 1997 is the innovative use of computational grid. Computational grid can help to get the solutions of complex models, considered intractable with ordinary computers, though at a significantly lower speed. But then it provides platform for researchers to carry out the state of art research with no big price tag. It is for this reason “Grid Computing” is called a poor man’s super computer.

The main requirements of efficient use of this technology are;

i) Logic to break the complex problem into numerous nearly independent segments which can be handled by individual nodes.
ii) Develop or use software which can achieve the above objective efficiently.

iii) The S/W should be portable, scalable & reliable.

Literature survey & review led to various options. For example there are platforms of the type Globus Toolkit, Legion, BOINC etc. But they are mainly based on Linux platform. As the majority of the computers available are windows based, it will be more convenient to develop a larger network of computers which will use the processing power of the computer to solve the complex problems. Alchemi.NET was one such option. So initially the work was carried out using this platform. During the research it was found that one could innovate through the option of multi threading a new S/W using an economic based policy. It was named as Nimble@ITCEcnoGrid.

Nimble@ITCEcnoGrid (The reason it is called Nimble as that it is very fast) is a novel homemade computational grid with the support of multithreading that works on windows operating system that computes the value of PI (\(\pi\)) up to 120 digits in milliseconds as compared to a more than half a minute by its competitor Alchemi.NET (An Open Source toolkit based on windows OS). Nimble@ITCEcnoGrid also have a novel feature of the Economic based policy which does not exist in Alchemi.NET framework. The economic based policy will encourage the users to participate in the grid as incentive will be give to them based on there computer usage.

Nimble@ITCEcnoGrid is more reliable than Alchemi.NET as the failure of one manager, sub manager does not affect the result as alternative managers & sub managers are available in the locations. There can be multiple managers at different locations and each manager can have multiple sub managers connected with it, which in turn connected with a group(can be variable in size) of executors independently.

Nimble@ITCEcnoGrid Framework (A Fast Grid with Inter-thread communication with Economic Based Policy) has been tested for the following applications & compared the performance with Alchemi.NET;

i) Computation of value of PI upto 120 decimal positions,

ii) Computation of the Momentum, Thermodynamics and Continuity equations , part of the first order Numerical Weather Prediction model,

iii) Determination of Prime Numbers.


v) Combined i) & ii), ii) & iii), i) & iv)

In all cases the results show significant improvement with respect to Alchemi.NET, Scalability with increase in number of executers & consistency. Its application for three different types parallelizable problems also show the versatility of the system. It may however be remembered that the Grid with Nimble provides excellent platform for research, but not necessarily for mission sensitive applications.

Keywords- Nimble@ITCEcnoGrid, Thermodynamics, Momentum, Continuity, Factorization, Web service, Manager, Sub Manager, Executor, Task Parallel Library.

1. INTRODUCTION

The processing power of the computers has been increasing exponentially keeping up the speed enunciated by Moore's law[1]. The requirement of the power & other infrastructure however keeps on increasing with still greater speed. This ever increasing demand can be met by distributing work load to more than one processor by using grid computing."

The Foster-Kesselman duo organized in 1997, at Argonne National Laboratory, a workshop entitled “Building a Computational Grid”. At this moment the term “Grid” was born. The workshop was followed in 1998 by the publication of the book “The Grid: Blueprint for a
New Computing Infrastructure” by Foster and Kesselman [42] themselves. For these reasons they are considered the fathers of the Grid.

1.1 HISTORY & OVERVIEW OF GRID

The term “Grid” was coined in the 1997 to denote a proposed distributed computing infrastructure for advanced science and engineering. The concept of Computational Grid has been inspired by the ‘electric power Grid’, in which a user could obtain electric power from any power station present on the electric Grid irrespective of its location, in an easy and reliable manner. Whenever additional electricity is required, just plug into a power Grid to access additional electricity on demand, similarly for computational resources plug into a Computational Grid to access additional computing power on demand using most economical resources [2] [3].The history of grid computing is shown in table 1.1.

Table:1.1 History to Grid Computing [43]

<table>
<thead>
<tr>
<th>Technology</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Networked Operating Systems</td>
<td>1979-81</td>
</tr>
<tr>
<td>Distributed operating systems</td>
<td>1988-91</td>
</tr>
<tr>
<td>Heterogeneous computing</td>
<td>1993-94</td>
</tr>
<tr>
<td>Parallel and distributed computing</td>
<td>1995-96</td>
</tr>
<tr>
<td>The Grid</td>
<td>1997</td>
</tr>
</tbody>
</table>

In Grid computing, Many independent computers is grouped together to solve a particular complex intensive & tedious problem. According to Foster-Kesselman [4] “ Grid Computing is a special type of parallel Computing which relies on complete computers (with onboard CPU, Power supply, network interface etc.) connected to the internet by conventional network interface, such as Ethernet”.

Grid computing is also defined as the coordinated resource sharing and problem solving in dynamic and multi-institutional Virtual Organizations[5]. A virtual organization can be composed of a group of individuals and/or institutions that come together to share resources with a common purpose.

2. LITERATURE REVIEW

Various types of the distributed systems and applications have been developed and are being used extensively in the real world. There are various types of distributed systems, such as Grids, Clusters [43], P2P (Peer-to-Peer) networks and so on. The major researches in the area of Grid Computing are being discussed here.

Iosup Alexandru [6] has suggested that the MTBF is high: around 12 minutes at grid level, 5 hours at the cluster level, and around 2 days per computing node. The duration of the computing node failures is 14 hours. Alexandru Iosup further find that when a failure occurs, it affects on average 10 or more computing nodes.

According to Alonso Javier [7], real time solution scenarios yet to be obtained. Grid applications comprise several components and web-services that make them highly prone to the occurrence of transient software failures and aging problems.

Brunelle J [8] suggested that the EGG (AN EXTENSIBLE AND ECONOMICS-INSPIRED OPEN GRID) project provides a vision and implementation of how heterogeneous computing
requirements will be supported within a single grid and a compelling reason to explain why computational grids will thrive.

According to Ranjan Rajiv [9], various kinds of solutions to grid resource discovery have been suggested, including the centralized and hierarchical information server approach. However, both of these approaches have serious limitations in regards to scalability, fault-tolerance and network congestion. The paper shows that there is a need of a reliable grid with improved technique of solving the problems.

Marcos Dias de Assuncao[10] suggested that Work in Grid has focused on interoperability, but not on the peering between Grids and its economic implications. The InterGrid needs to provide incentives for equivalent participation of individual Grids and resource providers. The Grids follow a restricted organizational model wherein a Virtual Organization (VO) is created for a specific collaboration and all interactions such as resource sharing are limited to within the VO. The work identifies and proposes architecture, mechanisms, and policies that allow the internetworking of Grids and allows Grids to grow in a similar manner as the Internet. The structure resulting from such internetworking between Grids as the InterGrid.

According to Chris Sosa[11], one reason Grid computing has lacked wide adoption because users are unwilling to suffer the cost of learning new paradigms or to modify applications to use Grid resources. There is certainly a need improved grid technology like inter thread communication in a multi threading environment that can enhance the grid capability.

According to Emmanuel Udoh,[12], the prediction of widespread adoption of grid technology has not materialized, while cloud technology is increasingly gaining acceptance. The growth of the enterprise grid, after its success in the research and academic communities, has been a gradual process. The cloud computing is gaining popularity but it is good in the commercial domain area. As far as the research area is concerned, Grid computing is the backbone of cloud computing.

Alexandru Iosup [13] suggested that Grids still pose many research challenges—among them, the high and variable job wait times. To continue evolving and tuning grids for production work, it’s important to understand the characteristics of entire grid workloads. Grids are collections of resources ranging from clusters to supercomputers. A technology in Grids and Clouds need to be exhausted to get good throughput.

According to M. Calin[14], grid computing networks are long term projects which finally bring undoubted benefits that can be measured through financial effect, scientific and social impact. However, they have an initial costly investment phase and the discussed worldwide examples reveal that both government agencies and private companies participated with money and equipment’s software in building an appropriate infrastructure. The development of the GRAI project follows this line of action.

Kunihiko Nishihashi [15] proposed a new method for large scale volume data rendering on a grid computing system, Dynamic Job-Scheduling using obstacle-flags. The proposed method generally performs better than the sequential job-scheduling: simulation results showed the reduction of elapsed time and the improvement of average number of agents utilized. The rendering progress of the proposed method was also smoother than the sequential job-scheduling.

According to S. S. Manvi, [16], grid computing is a concept, a network, a work in progress, part hype and part reality, and it is increasingly capturing the attention of the computing community. It uses clusters of personal computers, servers or other machines. They link together to tackle complex calculations. There can be a Gridnet in the future as we have the Internet today.

G. Murugesan[17] observed that resource management and scheduling plays a crucial role in achieving high utilization of resources in grid computing environments. Due to heterogeneity
of resources, scheduling an application is significantly complicated and challenging task in grid systems.

Aminul Haque [18] has suggested that Economic models are found efficient in managing heterogeneous computer resources such as storage, CPU and memory for grid computing. Commodity market, double auction and contract-net-protocol economic models have been widely discussed in the literature.

Yuan-Shun Dai[19] suggested that grid computing is a newly developed technology for complex systems with large-scale resource sharing, wide-area communication, and multi-institutional collaboration. It is hard to analyze and model the Grid reliability because of its largeness, complexity and stiffness. Therefore, this paper introduces the Grid computing technology, presents different types of failures in grid system.

According to Selvarni S[20], the goal of grid computing is to provide powerful computing abilities for complicated tasks by using all available and free computational resources. A suitable and efficient scheduling algorithm is needed to schedule user jobs to heterogeneous resources distributed on the grid. So scheduling is an important issue in a grid computing environment. According to Theo Ungerer [21], the Instruction-Level Parallelism found in a conventional instruction stream is limited. Studies show the limits of processor utilization even for today’s superscalar microprocessors. The main approach is the single chip multiprocessor and the multithreaded processor which optimizes the throughput of the multiprogramming workloads rather than the single treads-performance. The multithreaded processor is able to pursue two or more threads of control in parallel with the processor pipeline.

Chen Yen-Kuang [22] characterizes selected workloads of multimedia applications on current superscalar architectures, and then it characterizes the same workloads on Intel Hyper-Threading Technology. Our goal is to better explain the performance improvements that are possible in multimedia applications using Hyper-Threading Technology. Hyper-Threading Technology can increase the utilization of processor resources by 15 to 27.

Tian Xinmin [23] told that in the never-ending quest for higher performance, CPUs become faster and faster. Processor resources, however, are generally underutilized by many applications. Intel’s Hyper-Threading Technology is developed to resolve this issue. Processors enabled with Hyper-Threading technology can greatly improve the performance of applications with a high degree of parallelism. It has been shown that it is crucial to reach an optimal load balancing for an efficient implementation on Hyper-Threading Technology.

According to Wu Ming [24], the efforts to construct a national scale grid computing environment has brought unprecedented computing capacity. Exploiting this complex infrastructure requires efficient middleware to support the execution of a distributed application, composed of a set of subtasks, for best performance. This presents the challenge how to schedule these subtasks in shared heterogeneous systems. Current work has several limitations.

According to Agrawal Kunal [25], Multiprocessor scheduling in a shared multiprogramming environment is often structured as two-level scheduling, where a kernel level job scheduler allots processors to jobs and a user-level task scheduler schedules the work of a job on the allotted processors. For overall system efficiency, the task scheduler should also provide parallelism feedback to the job scheduler to avoid the situation where a job is allotted processors that it cannot use productively.

Kaufmann P [26] in his paper writes that numerical prediction model is used from 1994 to 2001. According to him the precipitation in the summer is very strong and variable and therefore it is very difficult to predict the weather in the summer rather than the winter. This reduces the performance of the model for the weather forecasting. He further discussed some case studies.
Vitart Frederic [27] in his paper suggested that Medium-range weather forecasting is essentially an atmospheric initial value problem. A monthly forecasting system based on 32-day coupled ocean–atmosphere integrations has been set up at ECMWF. This system has run routinely since March 2002 every 2 weeks, and 45 cases from March 2002 to December 2003 have been verified.

According to Chien Andrew [28], The exploitation of idle cycles on pervasive desktop PC systems offers the opportunity to increase the available computing power. However, for desktop PC distributed computing to be widely accepted within the enterprise, the systems must achieve high levels of efficiency, robustness, security, scalability, manageability, unobtrusiveness, and openness/ease of application integration. Entropia ceased commercial operations in 2004.

Foster Ian [29] [30] & Kesselman Carl[30] suggested that the Globus project is a multi-institutional research effort that seeks to enable the construction of computational grids providing pervasive, dependable, and consistent access to high-performance computational resources, despite geographical distribution of both resources and users. It is based on Linux operating system. The toolkit comprises a set of components that implement basic services for security, resource location, resource management, communication, etc.

Lewis.M [31] discussed the Legion project and is investigating issues relating to software architectures and base technologies for grid environments. In contrast to the Globus bag of services architecture, Legion is organized around an object-oriented model in which every component of the system is represented by an object.

Douglas Thain[32] investigated that Condor is a high-throughput computing environment whose goal is to deliver large amounts of computational capability over long periods of time (weeks or months), rather than peak capacity for limited time durations (hours or days). Condor is an open source high-throughput computing software framework for coarse-grained distributed parallelization of computationally intensive tasks.

According to Andeerson D.P [33], SETI@home uses computers in homes and offices around the world to analyze radio telescope signals. Its purpose is to analyze radio signals, searching for signs of extra terrestrial intelligence.

According to Poshtkohi Alireza [34], DotGrid is the first comprehensive Desktop Grid software utilizing the Microsoft’s .NET Framework in Windows-based environments and MONO.NET in Unix-class operating systems to operate. Crypt Engine architecture is one of sub-applications of DotGrid and is based on one of its models named hierarchical master-slaves topology.

According to Reddy, Shrinivas M.V [35], a two layered peer-to-peer middleware, Vishwa, handles reconfiguration of the application in the face of failure and system load. The two-layers, task management layer and reconfiguration layer, are used in conjunction by the applications to adapt and mask node failures. High performance applications have varied requirements in terms of CPU, memory, network bandwidth etc. Such techniques aim to smooth out access patterns and reduce the chance of transient overload.

Broberg James[36] suggested that traditional resource management techniques (resource allocation, admission control and scheduling) have been found to be inadequate for many shared Grid and distributed systems, that consist of autonomous and dynamic distributed resources contributed by multiple organizations. They provide no incentive for users to request resources judiciously and appropriately, and do not accurately capture the true value, importance and deadline of a user’s job.

Luther Akshay [37] [38] suggested that computational grids that couple geographically distributed resources are becoming the de-facto computing platform for solving large-scale problems in science, engineering, and commerce. Cross-platform support is provided via a web services interface and a flexible execution model supports dedicated and non-dedicated
(voluntary) execution of grid nodes. The comparison of Alchemi with other toolkits is given in table 2.1.

Alchemi [39] is an open source software framework that allows us to painlessly aggregate the computing power of networked machines into a virtual supercomputer (desktop grid) and to develop applications to run on the grid. Alchemi includes:

- The runtime machinery (Windows executable) to construct computational grids.
- A .NET API and tools to develop .NET grid applications and grid-enable legacy applications

Table 2.1: Comparison of Alchemi and some related Enterprise Grid Systems[40]

<table>
<thead>
<tr>
<th>Grid &amp; runtime environment</th>
<th>Grid Middleware</th>
<th>Model Level</th>
<th>Programming Languages</th>
<th>Web Services</th>
<th>Java Service</th>
<th>Hibernate</th>
<th>Grid FIFO</th>
<th>Grid Pool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alchemi</td>
<td>Yes</td>
<td>Low</td>
<td>C#, Win C</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Emора</td>
<td>No</td>
<td>Low</td>
<td>C#, Win C</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Gik@ITCEcnoGrid</td>
<td>Yes</td>
<td>High</td>
<td>C#, Win C, Java, Win 32</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

3. METHODOLOGY of Nimble@ITCEcnoGrid

Grid computing has emerged as a new paradigm for distributed computing. It enables scientists and engineering professionals to solve large scale computing problems. The proposed approach used the home made client server model architecture to construct the Grid toolkit namely “Nimble@ITCEcnoGrid”. There are Six original goals of the Nimble@ITCEcnoGrid:-

1) To do the evolution of the grid internally by improving the internal structure of the Grid Middleware by using the Inter Thread Communication.

2) To prove higher throughput and efficiency of the toolkit as compared to the existing popular alchemi.net toolkit by finding the value of PI (π) up to 120 decimal places after the decimal.

3) To solve intensive scientific calculations such as of Thermodynamics, Continuity and momentum using the first order model of weather forecasting.
4) To find economic based policy scheme as an incentive to the executors, who are dedicating their free cycles of processor to solve large scale computing problems.

5) To find the Prime Numbers up to 2000 numbers by using the concept of multithreading.

6) To find the factors of the numbers up to 200.

3.1 Requirements for Nimble@ITCEcnoGrid

The H/W & S/W used for the test case in Nimble@ITCEcnoGrid is

1. Core to Dual 3.0 GHz with 6MB Cache, Core i3 2.3 GHz, Core-to-Dual 1.88 GHz etc.
2. Network Interface Card
3. 8 port Network Switch
4. SQL Server 2008 R2
5. Visual Studio 2008 SP1 or VS 2010 Beta 2
6. Windows Server 2008 R2

3.1.1 First Step is to install Installation of Server 2008 [44]

3.1.2 Second Step is to Install SQL 2008 [45]

3.1.3 Third Step is to Install VS 2010 [46]

3.2 Basic Working Model of Nimble@ITCEcnoGrid

Nimble@ITCEcnoGrid is a web based Interface, which can be accessed & controlled by the Administrator. Nimble@ITCEcnoGrid Managers work at different locations. There can also be multiple managers at one location this is because, the failure of one manager at same location will not affect the work (e.g. If one manager at location Chandigarh stops working, another manager will take over and do the work) Data after calculation is sent to Nimble@ITCEcnoGrid Web Service, which will display the results.

There are a number of Executors connected to the Sub manager which are in turn connected to the Manager. For one manager there can be multiple sub managers and for one sub manager there can be multiple executors as shown in figure 3.1. The manager will assign the first assigned task to 1st idle available sub manager which further will distribute the work to executors attached with it. If another task arrives at the same time then it will be allotted to 2nd idle available sub manager by the manager which further will distribute the work to executors attached with it.

Executors after doing the required task will submit the result to the Sub Manager, which in turn after amalgamation of data will send results to the Manager. The manager will further send the results to the database of Nimble@ITCEcnoGrid web service and result will be displayed. More the number of executors, the less time the manager will take to produce results. The executors must work on real IP’s or LAN or VPN because if it gets disconnected in between for some reason, after reconnection the IP address will remain same. The participation time of the executors will get stored in the database and incentives can be given based on the time of participation in a grid.

3.3 Model of Nimble@ITCEcnoGrid

The model of Nimble@ITCEcnoGrid contains the following components

1) Manager
2) Sub Manager
3) Executors
4) Web Interface
3.3.1 Web Service
The web service is a software system design to support interoperable machine to machine interaction over the network. It is accessible to the authorized persons only. There are number of Managers connected with the web service. The information of all the managers is available in the database of the weather forecasting. When the Work arrives, Web Service allocates the work to the available managers.

![Figure 3.1: Model of Nimble@ITCEcnoGrid](image)

All the Status information about the managers is available to the administrator through a web service interface. The interface of the Nimble@ITCEcnoGrid web service is designed in .NET framework 3.5 which is easily upgradable to .NET framework 4.0.

Once the Manager collects the results from the sub manager, it will send the results to the Nimble@ITCEcnoGrid web service for display. There are options for calculation of the four equations namely:

1. Momentum for weather forecasting
2. Continuity for weather forecasting
3. Thermodynamics for weather forecasting
4. PI ($\pi$) Value calculation up to 120 digits/Prime Numbers for calculation from 1 to 2000 numbers.
5. Finding the factors of the number up to 200.

User can choose any number of available four options by selecting the check box and then clicking the Submit button to view the results after being calculated.
### 3.3.2 Manager
The most important work of the manager is the job allocation to sub manager and control. Nimble pool consists of Manager and arbitrary number of Sub managers and Executor attached with the sub manager. The Manager has the record of the Sub managers available in the pool of resources. There can be multiple Managers and the number of sub managers attached to a manager can also be multiplied. When the sub manager joins the pool of managers, their entry time is saved in the database.

The job of the manager is to allocate the job to the first available sub manager of the particular location. The sub manager is selected by the manager with the rule of first joined first allocated (FIFA) i.e. If more than one sub managers are available to do a current job, the job will be allocated to the sub manager who have firstly joined the pool.

The multiple managers attached to the web service provides the reliability i.e. If one manager fails then web service can allocate the job to some other available manager at that location. This is the prime benefit of this grid over the others including Alchemi.

In Alchemi, only one manager is there and if that manager fails then the whole process of calculation stops. Where as in the case of Nimble@ITCeGrid, if one manger fails at a particular location, then the job will be allocated to some other manager of that location or can be allocated to some other manager of some different location.

Microsoft’s Task Parallel library [47] is used for distributing tasks across the different computer cores using the concept of multi threading. TPL is the part of the system.threading namespace. One of the TPL advantage is that it automatically balances the load across the available sub managers e.g. suppose we run four tasks without TPL and we assign two to run on one CPU and two to run on second.

### 3.3.3 Sub Manager
Sub manager is connected with a manager. There can be multiple sub managers connected with one manager. The role of the sub manger is to accept the job from the manger and distribute it to the pool of available executors attached with it. It receives the results from the executors and then sends the result back to the manager.

Microsoft’s Task Parallel library is used for distributing tasks across the different computer cores using the concept of multi threading. It monitors the executors attached with it. If one executor fails to run the job, it will immediately shift the job to the available executor or will put it in a queue for the execution.

### 3.3.4 Executor
There can be a number of executors attached with the sub manager. Executors receive the job of the calculation of its sub manager and starts processing the job. After processing & Calculation, the executor returns the work to its sub manager which amalgamate the results and send the results to the manager. If one executor fails, the sub manager monitoring the executors will immediately transfer the pending job to another available executor or put in a queue.

The executors are used to run the following applications:-

a) **Weather Forecasting Equations**
According to the Kevin E Trenberth National Center for Atmospheric Research Boulder, Colorado USA [41], the Governing laws e.g. For the Atmosphere, the following equations are used as the first order model for the weather forecasting numerical calculations.

- **Momentum equations:**
  \[
  \frac{dV}{dt} = -\nabla p - 2\Omega \times V - gk + F + Dm
  \]

  Where \( \alpha = 1/\rho \) (\( \rho \) is density), \( p \) is pressure, \( \Omega \) is the rotation rate of the Earth, \( g \) is acceleration due to gravity (including effects of rotation), \( k \) is a unit vertical vector, \( F \) is friction and \( Dm \) is vertical diffusion of momentum.
Thermodynamic equation:
\[ \frac{dT}{dt} = \frac{Q}{cp} + \frac{(RT)}{p}\omega + DH \]
Where \( cp \) is the specific heat at constant pressure, \( R \) is the gas constant, \( \omega \) is vertical velocity, \( DH \) is the vertical diffusion of heat and \( Q = Q_{\text{rad}} + Q_{\text{con}} \) is internal heating from radiation and condensation/evaporation.

Continuity equations, e.g. For moisture (similar for other tracers):
\[ \frac{dq}{dt} = E - C + Dq \]
Where \( E \) is the evaporation, \( C \) is the condensation and \( Dq \) is the vertical diffusion of moisture.

Mainly the above defined equations give the weather forecasting basic equations. In this thesis the above equations are used to compute the value of the Momentum, Thermodynamic & Continuity equation.

b) Pi (\( \pi \)) value Calculation
Similarly for calculating the value of Pi (\( \pi \)) up to 120 digits a, the following procedure is adopted:

A method that is chosen is fairly simple and converges reasonably fast. It is based on the following formula:
\[ \frac{\pi}{4} = 4 \times \tan^{-1}(\frac{1}{5}) - \tan^{-1}(\frac{1}{239}) \]
This is Machin's formula [48] and is exact.

\( \tan^{-1}(\cdot) \) is the Inverse Tangent function, and Maclaurin series [49] is used to calculate it:
\[ \tan^{-1}(z) = z - \frac{z^3}{3} + \frac{z^5}{5} - \frac{z^7}{7} + \ldots \]
By including sufficiently many terms of this series, we can achieve any desired accuracy. To get 1,000,000 decimal places accuracy for \( \pi \), we need about 715,000 terms of the \( \tan^{-1}(1/5) \) series and about 210,000 terms of the \( \tan^{-1}(1/239) \) series. The program used to calculate 120 decimal places accuracy produces the result correctly & efficiently.

c) Prime Numbers/Factors of a number Calculation
Prime number is a special type of number, greater than one which is divisible by 1 or itself. The numbers \( \{2,3,5,7,11,13,17,19,23,29,31,37,\ldots\} \) are all prime numbers as these numbers are divisible by 1 or itself. Prime Numbers are used in Cryptography such as Public Key Cryptography as it is difficult to factorize large numbers into their prime number factors.

The number is checked if it is divisible by 1 or itself then it is a prime number. The above program logic gives the prime numbers up to 2000. 1\(^{\text{st}}\) executor computes the prime numbers from 1 to 500, 2\(^{\text{nd}}\) executor computes the prime number from 501 to 1000, 3\(^{\text{rd}}\) executor computes the prime number from 1001 to 1500 and so on. Similarly the factors of number from 1 to 200 can be calculated.

4. NIMBLE@ITCENOGRID RESULTS (Implemented on LAN)
Nimble@ITCEnogoGrid is a novel homemade toolkit developed to solve the equations of the 1\(^{\text{st}}\) order model of weather forecasting (having the equations of Momentum, Continuity & Thermodynamics), \( \Pi(\pi) \) value up to 120 digits, finding Prime Numbers from 1 to 2000 & factors of Number from 1 to 200. There are many test cases have been made to show its feature of reliability & efficiency. In the various test cases, the values of Momentum, Continuity, Thermodynamics, \( \Pi(\pi) \) & Prime number from 1 to 2000 numbers are calculated on grid individually and collectively.

4.1 Nimble@ITCEnogoGrid using Sub manager=1 Executor=1
It has 18 test cases in total. In the test case, the following configuration computers are used.
Manager Configuration : Core to Dual 3.0 GHz with 6MB Cache
Sub Manager Configuration : Core to Dual 3.0 GHz with 6MB Cache
Executor Configuration : Core i3 2.13 GHz  
No. of Manager : 01  
No. of Sub Manager: 01  
No. of Executors : 01  
The graphical form of results obtained is shown in figure 4.1.

Figure 4.1 : Time Taken (in ms) for calculations having 2 sub managers with 1 executor with each sub manager

4.2 Nimble@ITCEcnoGrid using Sub manager=2, Executor=1+1(Annexure ‘2’)

It has 18 test cases in total. In the test case, the following configuration computers are used. 
Manager Configuration : Core to Dual 3.0 GHz with 6MB Cache  
Sub Manager Configuration 1: Core to Dual 3.0 GHz with 6MB Cache  
Sub Manager Configuration 2: Core i3 2.13 GHz  
Executor Configuration : Core i3 2.13 GHz  
No. of Manager : 01  
No. of Sub Manager: 02 ( No 1 & No 2)  
No. of Executors : 02( one Attached With Sub Manager No 1) & (one Attached With Sub Manager No 2)  
There is 1 Manager, 2 Sub Managers and 2 Executor (1 executor is attached with each sub manager) as shown in figure 4.2. 
The graphical form of results obtained is shown in figure 4.2.
Figure 4.2: Time Taken (in ms) for calculations having 2 sub managers with 1 executor with each sub manager

The comparison between figure 4.1 & figure 4.2 in graphical form is shown in figure 4.3.

Figure 4.3: Comparing figure 4.1 & figure 4.2 in Graphical Form

As we compare the figure 4.1 relative to figure 4.2, it is find that the efficiency is increased in terms of less time taken i.e. it takes less time to do the calculations.
4.3 Nimble@ITCEcnoGrid using Sub manager=1 Executor=5
It has 18 test cases in total. In the test case, the following configuration computers are used.
Manager Configuration: Core to Dual 3.0 GHz with 6MB Cache
Sub Manager Configuration: Core to Dual 3.0 GHz with 6MB Cache
Executor Configuration: Core to Dual 3.0 GHz with 6MB Cache, Core i3 2.13GHz, Core-to-Dual 1.88GHz
No. of Manager :01
No. of Sub Manager:01
No. of Executors :05 (One core to dual 3.0 GHz, One core i3 2.13 GHz, 3 Core-to-dual 1.88 GHz)
The graphical form of results obtained is shown in figure 4.4.

![Figure 4.4: Time Taken (in ms) for calculations having Executors=5](image)

4.4 Nimble@ITCEcnoGrid using Sub manager=2 Executor=5+5
It has 18 test cases in total. In the test case, the following configuration computers are used.
Manager Configuration : Core to Dual 3.0 GHz with 6MB Cache
Sub Manager 1 Configuration : Core to Dual 3.0 GHz with 6MB Cache
Sub Manager 2 Configuration : Core to Dual 3.0 GHz with 6MB Cache
Executor Configuration : Core to Dual 3.0 GHz with 6MB Cache, Core i3 2.13GHz, Core-to-Dual 1.88GHz
No. of Manager :01
No. of Sub Manager :02
No. of Executors :10 (05 each with each sub manager with One core to dual 3.0 GHz, One i3 2.13 GHz, 3 Core-to-dual 1.88 GHz)
The graphical form of results is shown in figure 4.5.
Figure 4.5: Time Taken (in ms) for calculations having Executors=5 for sub manager 1 and Executors=5 for sub manager 2 collectively.

The comparison between figure 4.4 & 4.5 is shown in figure 4.6

Figure 4.6: Comparing figure 4.4 & figure 4.5 Graphically
The reason of less timing in the calculation of figure 4.6 is that the work is distributed to both the sub manages by the manager. Now ten executors are doing calculations simultaneously. If one sub manager fails, it will switch over the task to a second sub manager.

5. Economic Based Policy

The basic concept behind the Economic Based Policy is that any executor who is providing the computer to solve the complex & Intensive scientific calculations should get some incentive. This will encourage more and more users to be a part of a grid and a larger grid can be formed with this.

The figure 5.12 shows the amount earned by the executor by giving its free cycles to the grid. The rate is .02 Rs per Sec. The figure shows that IP no 192.168.12.253 has worked for 150 seconds and has earned Rs 3. Similarly 202.164.38.39 has worked for 9 seconds and has earned Rs 0.18. This will work on Real IP’s. The reason is that the IP number of the executor will not change and the incentive can be given according to the executors according to their dedicated time to sub manager.

The executor’s need real IP’s because their IP will remain the same and incentives can be given on this basis. Date wise, Month wise incentives of the executors can be easily generated in a crystal report in .rpt, .pdf, .doc,.xls formats. The crystal report of executors generated is shown in figure 5.1. Any positive numeric value can be give for finding the incentive.

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Executor ID</th>
<th>Executor Host</th>
<th>Start Date/Time</th>
<th>End Date/Time</th>
<th>Work Time (Sec)</th>
<th>Rate (Rs/Sec)</th>
<th>Total (Rs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>123456</td>
<td>192.168.12.253</td>
<td>01/01/2022 00:00</td>
<td>01/01/2022 23:59</td>
<td>150</td>
<td>0.02</td>
<td>3.00</td>
</tr>
<tr>
<td>2</td>
<td>654321</td>
<td>202.164.38.39</td>
<td>01/01/2022 00:00</td>
<td>01/01/2022 23:59</td>
<td>9</td>
<td>0.02</td>
<td>0.18</td>
</tr>
</tbody>
</table>
6. CONCLUSION

This thesis proposed a computational InterGrid (Nimble@ITCEcnoGrid) having distributed resources based on .NET framework to solve the first order weather forecasting model equations, Pi (π) value up to 120 digit places & finding Prime Numbers from 1 to 2000. Nimble@ITCEcnoGrid is having inter-thread connectivity which does not exist in any kind of Windows/Linux based Grid toolkits. An economic based resources allocation policy has been developed which does not exist in alchemi.net toolkit.

The present research work also compares Nimble@ITCEcnoGrid framework with the Alchemi.NET framework to illustrate the superiority over the later, to promote the grid oriented research and having the collaboration with the multi organizations. The results clearly indicate that the Nimble@ITCEcnoGrid framework model can be used as a good future scope for solving the complex scientific real time problems like Climate Prediction, Protein Folding etc.

From the various results presented in the section 4.1, 4.2, 4.3, 4.4, following significant conclusion can be derived:

i) Grid computing can be used to solve complex problem using free cycles of processors in shared mode.
ii) The home grown Nimble@ITCEcnoGrid is consistently superior to the Alchemi.
iii) Efficient use of the system has been demonstrated by using the system for six different types of the problems individually & in combination.
iv) The system developed & presented in this thesis shows that it is scalable, reliable & robust for different types of applications

6.1 Limitations
As the real time problems need a time bound & higher precision calculations, Nimble@ITCEcnoGrid may be used in a Research domain and not independently in the operational domain until fast network like fiber optics is available having transfer rate in gigabits.
The results have been taken on the cleanest systems after proper formatting the systems and with minimum required software. The results may differ somewhat if the machines used are highly loaded (Doing multiple tasks like virus scan etc.), if the network have congestion, if some heavy software are installed and if the results are taken in the peak hour of traffic.
Microsoft’s Task Parallel library is used for distributing tasks across the different sub managers & executors. The TPL is the inbuilt technique of distributing the work in .NET. So, there is a limitation in load balancing where processes have some sequencing requirements.

6.2 Future Scope
The future scope of the research is to implement the Nimble@ITCEcnoGrid Model in the real time problems like Climate Prediction, Protein Folding etc with high speed networks. Some new scheduler algorithm may be developed that will work on Managers and Sub Managers to distribute the work more frequently and efficiently that will lead to more frequent result.
Economic Based policy can be further improved if the incentives can be given on the basis of the classification of the PC machines (i.e. the more powerful machine gets more incentive than the slower one).
Microsoft’s Task Parallel library is used for distributing tasks across the different computer cores using the concept of multi threading. A better and optimized algorithm/technique may be applied in the future so that the distribution of work may be done properly.

REFERENCES


[22] Chen Yen-Kuang, Holliman Matthew, Debes Eric (2002),” Media Applications on Hyper-Threading Technology” Intel Technology Journal Q1, Vol. 6 Issue 1


