Two Day Conference

Linux Thane 2006

November 24 - 25, 2006

Venue
Thorale Bajirao Peshwe Sabhagruha
'Jnanadweepa', Thane College Campus,
Chendani, Bunder Road, Thane (W) 400 601.

Editorial Committee

Abhijeet A. Kale
Rajgopal S. Iyer
Dalvir J. Reel

Organized by
Department of Information Technology
Vidya Prasarak Mandal's
B.N. Bandodkar College of Science
'Jnanadweepa', Chendani, Bunder Road,
Thane (W) - 400601, Maharashtra

Advisory Committee

Dr. V. V. Bedekar	Chairman, Vidya Prasarak Mandal, Thane
Dr. (Mrs.) M. K. Pejaver	Principal, B.N. Bandodkar College of Science, Thane
Mr. D. K. Nayak	Principal, V. P. M.'s Polytechnic, Thane
Mr. P. L. Vaidya	Industrial Consultant
Mr. A. A. Kale	In-charge, IT Department, B.N. Bandodkar College of Science, Thane

Program Committee

Dr. (Mrs.) M. K. Pejaver	Convener
Mr. A. A. Kale	Organizing Secretary
Mrs. A. D. Gangal	Member
Mrs. B. B. Bhole	Member
Mr. D. J. Reel	Member
Mr. R. S. Iyer	Member
Ms. A. A. Gadre	Member

Chairman's Note

World over India is looked upon with respect as a IT powerhouse. But for it to become IT Superpower the technology needs to percolate down to the masses. We are witnessing the impact of Telecom revolution in the urban and rural areas. With 8% GDP growth, the next wave would be of huge adoption of computing technology across the country. When talking about IT one cannot miss the cost factor.

Educational institutions, on their part, should spread awareness and equip the students with the current knowledge about the available technology and happenings in the Industry to give their students an edge. In India apart from research institutes, commercial organisations like Life insurance Corporation of India, Indiabulls.com, etc are adapting Linux.

In India, if we want to increase computer density in schools and colleges, we need to offer computing environment which our educational institutes can afford. Open source operating system can achieve help this goal without compromising any quality. Our universities and other government departments, while designing syllabus and other IT guidelines must advocate for Open source. I am sure this seminar will help policy makers in this direction.

Vidya Prasarak Mandal, on it's part, has deployed Linux based Thin client network with around 100 thin clinets catering to over 200 stakeholders in our management institute for day to day use. Though there were some teething problems, the end users started using it and are in fact quite happy with the performance, stability and privacy it provides.

It gives me great pleasure that Information Technology Department of B. N. Bandodkar College of Science is going to conduct a Two-day conference LinuxThane 2006. It has already conducted Two pre-conference workshops for B.Sc. (IT) & M.Sc. (IT) students and Lecturers of Mumbai University.

I wish them all the best for this conference and may this be the first in a series in coming years.

Dr. Vijay V. Bedekar
Chairman
Vidya Prasarak Mandal, Thane

Convener's Note

It gives me a great pleasure to give the proceedings of the National conference on 'Linux Thane 2006' in your hands as the Convener of the conference.

Though I am not from the field of I.T. being the Principal of the college I felt that some Vigorous discussion on this topic is needed as our vision is Imparting Quality Education in Science. Since we are committed to quality education to students and the community we believe to help the students to go beyond syllabus and interact with the scholars of repute from the field.

Conferences and Seminars are right platforms for the same, where every year we have a wide discussion on selected topic. We have to announce our national conferences one year in advance. We hold two workshops prior to the conference, where students and researches are exposed to the theme of the topic. This creates awareness among the students who look forward to research as one of the carrier option.

For LinuxThane 2006 also we conducted two workshops, end result of it is my two S.Y.B.Sc.I.T. Students are presenting their research paper in this Conference.

As a Convener of the Conference and Principal of the college I am proud of my I.T. Staff and Students. I appreciate the full support of our Vidya Prasarak Mandal, my teacher colleagues, non-teaching Staff, the students of our college and all participants.

I am specifically thankful to all the supporters who extended the hands of help to Mr. Abhijeet Kale, our organizing Secretary.

We value your opinion about this conference.

Dr Madhuri Pejavaer
Principal & Convener
B.N.Bandodkar College of Science, Thane.

Foreward

The landscape of computer software keeps changing day by day. The requirements of the users too keep increasing day by day. Though software, per se, can be made to do many things, some of the real world issues that keep popping are: Cost, Reliability, Support, Longevity, and Security etc. Individuals, organizations - whether private, academic, government, enterprises - have to grapple with these issues as usage of Computers keeps increasing day by day. There is software available for many different purposes. Often the relevant software is used to perform a task or some part of the overall work.

One very important thing to be remembered is that when one pays for software, they pay a license fee. In other words, one cannot own Proprietary software. They are legally a lessee. Think of it like one time rent (or yearly depending upon the License agreement). Most often individual desktop purchasers are not aware that they are perpetuating piracy when they ask their hardware vendor to "preinstall" many software without paying for the license in case of proprietary software. Some organizations, either due to pure and simple ignorance or to save costs, expect the hardware vendor to "preinstall" certain software without paying the license fee to the proprietors of the software. Often these licenses are very restrictive when it comes to redistribution, transfer etc.

FLOSS (Free / Open Source Software) on the other hand, provides an alternative to this proprietary model. The software is liberally licensed to grant the right of users to study, change, and improve its design through the availability of its source code and the right to distribute the changed program. FLOSS is being worked upon and continuously by many people around the clock, around the globe. The results have been tremendous, so much so that Harvard Business Review carried an article (How Toyota and Linux Keep Collaboration Simple 8/1/2005).

Recently Kerala government decided to deploy FLOSS in all state run schools with the help of Free Software Foundation India and many open source enthusiasts and practitioners. Many academic institutions like IIT Mumbai, C-DAC uses Linux extensively for its working. Apache, a web server, hosts almost 60 percent of the websites.

Though initial cost is an important factor in any IT Deployment, there are many more aspects to be studied in detail so that whatever is learned can be applied elsewhere fruitfully and effectively. This is especially true for academic institutions, which have to give exposure to the alternatives available in the marketplace.

Keeping the above in mind, VPM's B. N. Bandodkar College of Science is organizing a seminar "LinuxThane 2006" on 24th and 25th November 2006 The motto of Vidya Prasarak Mandal "prajvaalito jnaanamayaH pradeepaH" supported by conference LinuxThane 2006. It purports to spread awareness and explore some of the areas where work in the open source paradigm is being carried out.

Abhijeet A. Kale
Organizing Secretary

Two Day Conference

Linux Thane 2006

November 24 - 25, 2006

PROGRAMME

1st Day : Friday, November 24, 2006

09.00 Hrs. to 09.30 Hrs.	Registration, Tea & Snacks
09.30 Hrs. to 11.00 Hrs.	**Inaugural Function** **Chief Guest :** Mr. Shankar Iyer, Head, Global Learning Services, RedHat, India
	Guest of Honor : Prof. Sivakumar Head, Computer Science and Engineering Department Indian Institute of Technology, Mumbai
	Invitte Mr. Zia Aquib Executive Director, CDAC, Juhu, Mumbai
11.00 Hrs. to 12.00 Hrs.	Prof. Sivakumar (Free/Open Source Software in Engineering Curriculum)

Session I :

12.00 Hrs. to 12.30 Hrs.	Mr. Siji Sunny (Migrating IndiX from Xfree 86 to Xorg)
12.30 Hrs. to 13.30 Hrs.	**Lunch Break**
13.30 Hrs. to 14.00 Hrs.	Mr. Anal Haque Warsi (Another 'world' is possible. aanatabharati)

Session II :

14.00 Hrs. to 15.00 Hrs	Demo by SUSE-Linux
15.00 Hrs. to 15.30 Hrs.	**Tea Break**

Session III :

15.30 Hrs. to 16.30 Hrs.	Prof. Hiren Dand, Prof. Sagar Patil, Prof. Santosh Soni (PVM and MPI : A Comprehensive Study and Comparison)

2nd Day : Saturday, November 25, 2006

09.00 Hrs. to 09.30 Hrs.   Tea & Snacks

Session IV :
09.30 Hrs to 10.00 Hrs   Mr. Abhijeet Kale
                                    (Experimentation of Linux on VPM Campus)
10.00 Hrs to 10.30 Hrs   Ms. Priyanka Kadam and
                                    Ms. Yashada Naik (S.Y.I.T. Students)
                                    (Open source for Library)
10.30 Hrs. to 11.00 Hrs.   Prof. Verma
                                    (Cost of Software Corporate Users Percpective)

Session V :
11.00 Hrs. to 11.30 Hrs.   Mr. Arvind Sharma,
                                    Root Technologies
11.30 Hrs. to 12.30 Hrs.   Linux Pune Interactive Session
12.30 Hrs. to 13.30 Hrs.   **Lunch Break**

Session VI :
13.30 Hrs. to 15.00 Hrs.   Prof. Nagarjuna G.
                                    (Swatantra Software)

15.00 Hrs. to 15.30 Hrs.   **Tea Break**
15.30 Hrs. To 16.30 Hrs.   Valedictory Function
                                    Chief Guest : Prof. Nagarjuna G.
                                    Chairman, Free Software Foundation of
                                    India Scientist,
                                    Tata Institute of Fundamental Research, Mumbai

C O N T E N T S

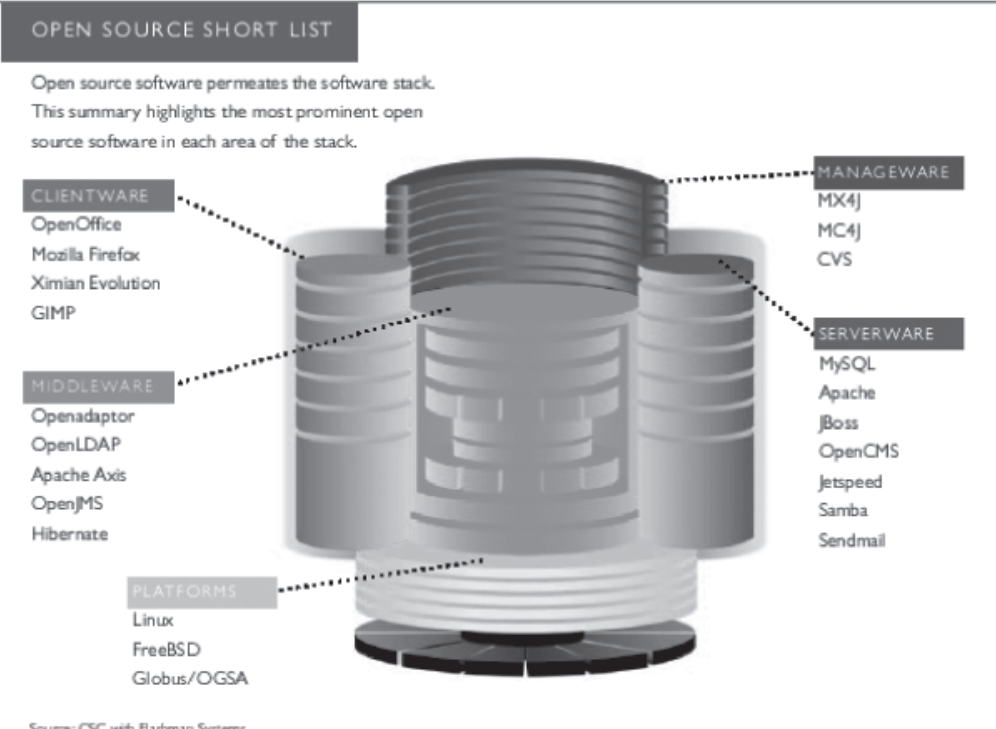# Free/Open Source Software in Engineering Curriculum

G. Sivakumar

Computer Science and Engineering
IIT Bombay
siva@iitb.ac.in

Outline

- What is Free/Open Source?
- Why FOSS for Education?
- A Taste of FOSS in Engineering Domains

FOSS Software

OPEN SOURCE SHORT LIST

Open source software permeates the software stack. This summary highlights the most prominent open source software in each area of the stack.

CLIENTWARE
OpenOffice
Mozilla Firefox
Ximian Evolution
GIMP

MIDDLEWARE
Openadaptor
OpenLDAP
Apache Axis
OpenJMS
Hibernate

PLATFORMS
Linux
FreeBSD
Globus/OGSA

MANAGEWARE
MX4J
MC4J
CVS

SERVERWARE
MySQL
Apache
JBoss
OpenCMS
Jetspeed
Samba
Sendmail

Source: CSC with Flashmap Systems

Interesting Examples

- Google (millions of searches every hour)
- Apache (70% share)
- Mozilla Firefox
- Open Office
- Sendmail
- Postgres
- sourceforge.net
- ...

Economist Innovation Award 2004

Award Criteria

"Tonight's awards recognise top innovators whose work has both driven progress in their particular fields and contributed more widely to global social and economic prosperity," said Standage. "The Economist is proud to recognise and thank them for their achievements."

Computing, Linux: Linus Torvalds, Fellow, Open Source Development Lab.

Torvalds originated Linux in 1991 as a 21-year-old computer science student at the University of Helsinki, Finland. Dissatisfied with the MS-DOS (and early Windows) operating system standard prevalent on PCs, Torvalds made Linux freely available for downloading, releasing the source code so that people with knowledge of computer programming could modify Linux to suit

Economist Innovation Award 2004
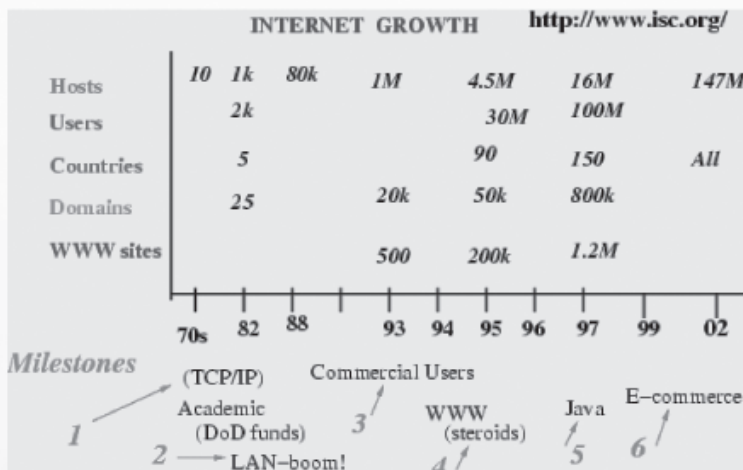
Award Criteria

"Tonight's awards recognise top innovators whose work has both driven progress in their particular fields and contributed more widely to global social and economic prosperity," said Standage. "The Economist is proud to recognise and thank them for their achievements."

Computing, Linux: Linus Torvalds, Fellow, Open Source Development Lab.

Torvalds originated Linux in 1991 as a 21-year-old computer science student at the University of Helsinki, Finland. Dissatisfied with the MS-DOS (and early Windows) operating system standard prevalent on PCs, Torvalds made Linux freely available for downloading, releasing the source code so that people with knowledge of computer programming could modify Linux to suit their own needs. The software created a huge following, eventually

Internet's Growth and Charter

| INTERNET GROWTH | http://www.isc.org/ |

	10	1k	80k	1M		4.5M	16M		147M
Hosts	10	1k	80k	1M		4.5M	16M		147M
Users		2k				30M	100M		
Countries	5					90	150		All
Domains	25			20k		50k	800k		
WWW sites				500		200k	1.2M		

70s 82 88 93 94 95 96 97 99 02

Milestones

(TCP/IP)   Commercial Users

Academic   *3*   WWW   Java   E-commerce
(DoD funds)   (steroids)

1   *2* → LAN-boom!   *4*   *5*   *6*

Information **AnyTime, AnyWhere, AnyForm, AnyDevice, …**
WebTone like DialTone

Internet Engineering Task Force

I E T F

RFC 2026

The Internet, a loosely-organized international collaboration of autonomous, interconnected networks, supports host-to-host communication through voluntary adherence to open protocols and procedures defined by Internet Standards.

Overiew

The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. It is open to any interested individual.

G. Sivakumar   Computer Science and Engineering IIT Bombay siva@iitb.ac.in

Free/Open Source Software in Engineering Curriculum

Why Open Standards?

1. **Availability**
   For all to read and implement.

2. **Maximize End-User Choice**
   Fair, competitive market. No lock-in to a particular vendor.

3. **No Royalty**
   Free for all to implement, with no royalty or fee. Certification may involve a fee.

4. **No Discrimination**
   Do not favor one implementor over another for any reason

5. **Extension or Subset**
   However, certification organizations may place requirements upon extensions.

6. **Protection against Predatory Practices**
   Embrace and enhance!

Free Software

http://fsf.org.in/ Richard M. Stallman

Free software

is a matter of freedom, not cost. It is a matter of liberty, not price. The word *free* in free software has a similar meaning as in free speech, free people and free country ... Think of free software as software which is free of encumbrances, not necessarily free of cost. Think of it as *swatantra* software.

Degrees of Freedom

1. The freedom to run the program, for any purpose
2. The freedom to study how the program works, and adapt it to your needs
3. The freedom to redistribute copies so you can help your neighbor
4. The freedom to improve the program, and release your improvements to the public so that the whole community benefits

Free Software

http://fsf.org.in/ Richard M. Stallman

Free software

is a matter of freedom, not cost. It is a matter of liberty, not price. The word *free* in free software has a similar meaning as in free speech, free people and free country ... Think of free software as software which is free of encumbrances, not necessarily free of cost. Think of it as *swatantra* software.

Degrees of Freedom

1. The freedom to run the program, for any purpose
2. The freedom to study how the program works, and adapt it to your needs
3. The freedom to redistribute copies so you can help your neighbor
4. The freedom to improve the program, and release your improvements to the public so that the whole community benefits

G. Sivakumar   Computer Science and Engineering IIT Bombay siva@iitb.ac.in
Free/Open Source Software in Engineering Curriculum

Free Software

http://fsf.org.in/ Richard M. Stallman

Free software

is a matter of freedom, not cost. It is a matter of liberty, not price. The word *free* in free software has a similar meaning as in free speech, free people and free country ... Think of free software as software which is free of encumbrances, not necessarily free of cost. Think of it as *swatantra* software.

Degrees of Freedom

1. The freedom to run the program, for any purpose
2. The freedom to study how the program works, and adapt it to your needs
3. The freedom to redistribute copies so you can help your neighbor
4. The freedom to improve the program, and release your improvements to the public so that the whole community benefits

G. Sivakumar   Computer Science and Engineering IIT Bombay siva@iitb.ac.in
Free/Open Source Software in Engineering Curriculum

FSF and GPL

The Free Software Foundation

Free software is a matter of liberty not price. You should think of "free" as in "free speech".

The Free Software Foundation (FSF), established in 1985, is dedicated to promoting computer users' rights to programs. The FSF promotes the development and use of free software, particularly the GNU operating syste helps to spread awareness of the ethical and political issues surrounding freedom in the use of software.

Here are the FSF's current projects.

GNU

FSF remains the primary sponsor of the GNU Project. In addition to the services provided by Savannah for GNU projects, FSF provides development systems for GNU software maintainers, including full email and shell services. We are committed to furthering the development of the GNU Operating System and enabling volunteers to easily contribute to that work.

Free Software Licensing and Compliance Lab

The commencement of the GNU project in 1984, with its goal to give users freedom, required the establishment of new distribution terms that would prevent the project being turned into proprietary software. The method used was Copyleft and the resulting license was called the GNU General Public License (GNU GPL). Today the GNU GPL is the most widely used Free Software license, and as its author, the FSF works to help the wider community use and comprehend it.

Free Software Directory

The Free Software Directory was started in September 1999 to catalog all useful free software that runs under free operating systems. The Directory contains over 3,000 entries. Recently, FSF has developed a partnership with UNESCO to combine our Free Software Directory with UNESCO's Free

G. Sivakumar   Computer Science and Engineering IIT Bombay siva@iitb.ac.in
Free/Open Source Software in Engineering Curriculum

Open Source Definition

From http://www.opensource.org (Author: Bruce Perens)

1. Free Redistribution
2. Source Code
3. Derived Works
4. Integrity of The Author
5. No Discrimination Against Persons or Groups
6. Distribution of License
7. License Must Not Be Specific to a Product
8. License Must Not Restrict Other Software
9. License Must Be Technology-Neutral

Open Source Licencing

open source

.:: License Index ::.

License Approval Process
License Information

*Academic Free License
*Apache Software License
*Apache License, 2.0
*Apple Public Source License
*Artistic license
*Attribution Assurance Licenses
*BSD license
*Common Public License
*CUA Office Public License Version 1.0
*EU DataGrid Software License
*Eclipse Public License
*Eiffel Forum License
*Eiffel Forum License V2.0
*Entessa Public License

FOSS for India

Indian president calls for open source in defense
Published: July 7, 2004, 10:03 AM PDT

By Dinesh C. Sharma
Special to CNET News.com

In another public-sector boost to open-source software, Indian President A.P.J. Abdul Kalam called for his country's military to use such nonproprietary technology to ward off cybersecurity threats.

"Software maintenance and software upgrade is an important issue for defense," Kalam said at a meeting of Indian Navy's Weapons and Electronic System Engineering Establishment in New Delhi last week.

Without naming any proprietary software products, the president asked defense engineers to develop and implement on open platforms. "Even though the required software for the equipment could be developed by the private industry, it is essential that the technical know-how and the architecture is fully available with these services for ensuring provision of lifetime support for the software which may or may not be forthcoming from the

G. Sivakumar   Computer Science and Engineering IIT Bombay siva@iitb.ac.in
Free/Open Source Software in Engineering Curriculum

Freeduc in Schools

Freeduc is a Live CD with wealth of FOSS for school kids.

- What does it cost to produce?
- How to distribute? (Postman: once in 3 months!)
- What expertise does it need?

What innovation is needed for India?

- Localization!
- Relevant content.

Who can/will do this innovation?
Compare with proprietary software!

G. Sivakumar   Computer Science and Engineering IIT Bombay siva@iitb.ac.in
Free/Open Source Software in Engineering Curriculum

Information Footpath

Super Highways are good, but why wait for that?

Freeduc is just one example.

eMoviX:

a micro Linux distro meant to be embedded in a CD together with all video/audio files you want, so that the CD will be able to boot and automagically play all files;

MoviX:

A mini CD Linux distro able to boot directly from CD and load in RAM a console interface to MPlayer. From the interface you can easily play DVDs, VCDs, audio/video files Audio CD, internet radios, TV, you name it!

Supported formats- AVI, MPG, QuickTime, MP3, OGG/VORBIS and a few others. See http://movix.sourceforge.net/

Easy, cost effective way to distribute information!

Static vs. Dynamic content

FOSS and Scientific Method

Sharing data and results is the foundation!

Way of Genius

If I have seen further [than others] it is by standing on the shoulders of giants... Issac Newton

Results are accepted only if they can be replicated! (sharing)

Open Access is a must!

Why FOSS in Academia

Tamil Proverb

What has been learned is like a fistful of sand, what remains is like the whole earth!

Solution?

Giving a scholar access only to *raw information* is like giving only seeds to a *hungry man*.

Way Forward?

Giving a student access only to *executable code* is like giving only *cooked rice* to a *farmer*.

How can *FOSS* close this gap?
Students move from being mere *users/consumers* to *producers*.
Great Empowerment!

Why FOSS in Academia

Tamil Proverb

What has been learned is like a fistful of sand, what remains is like the whole earth!

Solution?

Giving a scholar access only to *raw information* is like giving only seeds to a *hungry man*.

Way Forward?

Giving a student access only to *executable code* is like giving only *cooked rice* to a *farmer*.

How can *FOSS* close this gap?
Students move from being mere *users/consumers* to *producers*.
Great Empowerment!

Information Hierarchy

Open Access Journals

What is open access publishing?

1. Free and unrestricted online access to the research literature and databases

2. Users are licensed to download, print, copy, redistribute, and use

3. Author retains copyright and the right to be acknowledged

4. Papers are deposited in a public database that allows sophisticated searches (such as PubMedCentral)

5. (Bethesda Principles, April 2003)

Why is open access important?

1. Maximum impact for authors
   access to the largest possible audience
2. New ways to access and use literature
   full-text searching and mining (e.g. Google Scholar)
3. Greatly expanded access to research
   for scientists, educators, physicians, the public

Economic analysis at
http://www.wellcome.ac.uk/doc_WTD003181.html

Why don't researchers know or care?

Publishers £ Profit Shareholders & Societies

Free

£ Funders mission?

Free

Researchers Libraries

£ £

Public funders & HEFCE

The Wellcome Trust

Open Access not only for Consumers!

Wikipedia - Mozilla Firefox

ew   Go   Bookmarks   Tools   Help   W http://wikipedia.org/

WIKIPEDIA

English
The Free Encyclopedia
533 000+ articles

Deutsch
Die freie Enzyklopädie
225 000+ Artikel

日本語
フリー百科事典
110 000+ 記事

Français
L'encyclopédie libre
100 000+ articles

Svenska
Den fria encyklopedin
71 000+ artiklar

Polski
Wolna Encyklopedia
64 000+ haseł

Nederlands
De vrije encyclopedie
66 000+ artikelen

Español
La enciclopedia libre
46 000+ artículos

Italiano
L'enciclopedia libera
40 000+ articoli
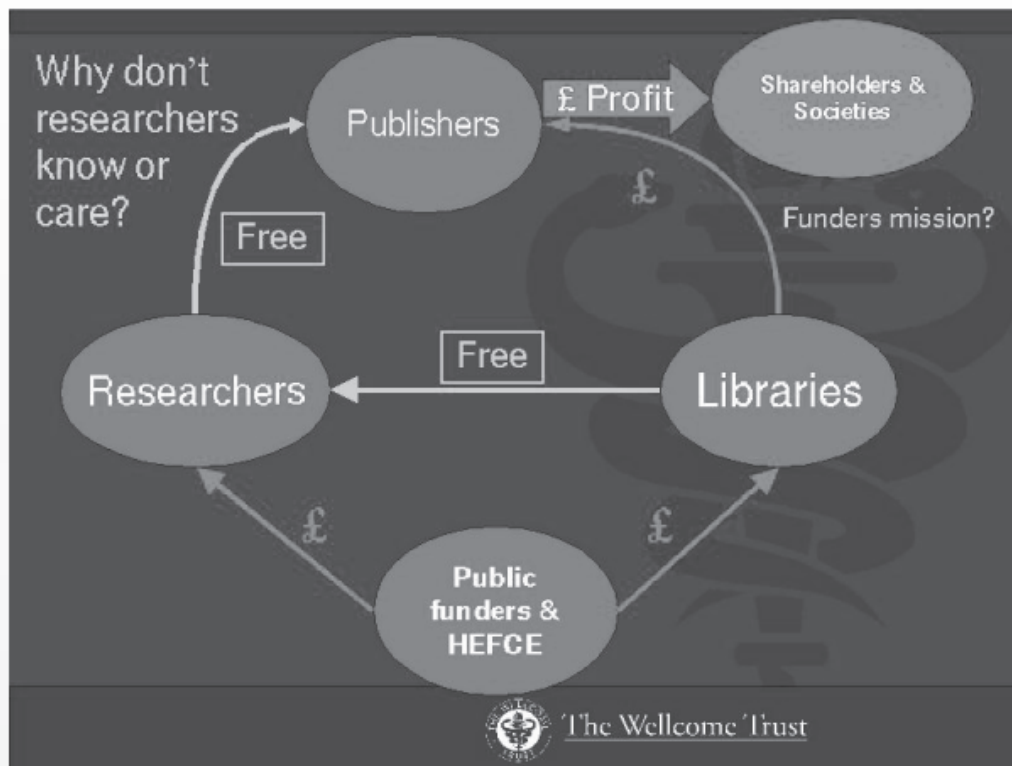
Português
A enciclopédia livre
37 000+ artigos

G. Sivakumar   Computer Science and Engineering   IIT Bombay   siva@iitb.ac.in

Free/Open Source Software in Engineering Curriculum

Wikipedia

Main Page - Wikipedia, the free encyclopedia - Mozilla Firefox

File   Edit   View   Go   Bookmarks   Tools   Help   W http://en.wikipedia.org/wiki/Main_Page

article   discussion   view source   history

Main Page

From Wikipedia, the free encyclopedia.

Welcome to Wikipedia, the free-content encyclopedia that anyone can edit.
In this English version, started in 2001, we are currently working on 559341 articles.

CULTURE | GEOGRAPHY | HISTORY | LIFE | MATHEMATICS | SCIENCE | SOCIETY | TECHNOLOGY
Browse Wikipedia · Article overviews · Alphabetical index · Other category schemes

Today's featured article

Louis Riel was a Canadian politician and leader of the Métis people of western Canada. He led two resistance movements against the Canadian government that sought to preserve Métis rights and culture as their homelands came under the Canadian sphere of influence. During the first, the Red River Rebellion, the provisional government established by Riel ultimately negotiated the terms under which the modern province of Manitoba entered the Canadian Confederation. He was forced into exile as a result of the controversial execution of Thomas Scott, but in 1884 he returned to what is now the province of Saskatchewan to participate in the North-West Rebellion of 1885. It ended in his arrest, trial and eventual execution for treason. Riel was viewed sympathetically in francophone regions of Canada, and his execution has had a lasting influence on relations between the province of Quebec and English-speaking Canada. Whether he is seen as a *de facto* Father of Confederation or as a traitor, he remains one of the most complex, controversial and ultimately tragic figures in the history of Canada.

Recently featured: Brian Close – Diamond – Samantha Smith

Archive – By email – **More featured articles...**

In the news
- A planned new N3 motorway past the Iri mediaeval Hill of Tara is approved, raising international outcry from environmentalist historians and Celtic scholars.
- The Sellafield nuclear plant's Thorp repro Cumbria, UK, is closed down due to a 20-t nitric acid.
- John Convers and 87 other members of George W. Bush about secret 2002 agreem invade Iraq.
- The wreckage of NASA's *Mars Polar Land* have been found on Mars in images from t

Wikinews · **Recent deaths**

Did you know...
From Wikipedia's newest articles:
...that *Starry Night Over the Rhone*, paint

G. Sivakumar   Computer Science and Engineering   IIT Bombay   siva@iitb.ac.in

Free/Open Source Software in Engineering Curriculum

Wikipedia - Wikipedia, the free encyclopedia - Mozilla Firefox

File Edit View Go Bookmarks Tools Help W http://en.wikipedia.org/wiki/Wikipedia

article | discussion | **edit this page** | history

Wikipedia

From Wikipedia, the free encyclopedia.

Wikipedia is a Web-based, free-content encyclopedia that is written collaboratively by volunteers. It consists o editions sponsored by the non-profit Wikimedia Foundation. Entries on traditional encyclopedic topics exist alor and current events topics. Its purpose is to create and distribute, worldwide, a free encyclopedia in as many lar one of the most popular reference sites on the Web[1], receiving around 60 million hits per day.

Wikipedia contains approximately 1.5 million articles, more than 500,000 of which are in its English language ed German language and more than 100,000 each in Japanese and French. It began as a complement to the expe 2001. Having steadily risen in popularity,[2] it has spawned several conceptually related sister projects such as Its articles are edited by volunteers in wiki fashion, meaning articles are subject to change by nearly anyone. W policy of "neutral point of view." Under this, the views presented by notable persons or literature are summarize an objective truth. Because of its open nature, vandalism and inaccuracy are problems in Wikipedia.

Wikipedia's status as a reference work has been controversial. It has received praise for being free, editable, ar for a perceived lack of accountability and authority when compared with traditional encyclopedias, systemic bia been cited by the mass media and academia. Wikipedia's articles are available under the GNU Free Document distributed on compact discs, and many of its other editions are mirrored or have been forked by websites.

Contents [hide]
1 Characteristics
  1.1 Free-content
  1.2 Language editions
2 Editing
  2.1 Policies
  2.2 Authors
3 Evaluations

Why no Indian Languages?

Wikipedia - Wikipedia, the free encyclopedia - Mozilla Firefox

File Edit View Go Bookmarks Tools Help W http://en.wikipedia.org/wiki/Wikipedia

Wikipedia as a source as a source

Language editions

Wikipedia encompassed 92 "active" language editions in March 2005.[8] Its five largest editions were, in descending order English, German, Japanese, French and Swedish. In total, Wikipedia contained 195 language editions of varying states wit combined 1.5 million articles.[9]

Language editions operate independently of one another. Editions are not bound to the content of other language editio and are only held to global policies such as "neutral point of view". Articles and images are nonetheless shared between Wikipedia editions, the former through pages to request translations organized on many of the larger language editions, the latter through the Wikimedia Commons repository. Translated articles represent only a small portion of articles in any edition.[10]

The following is the list of major editions in the order of article number in May 2005.[11]

1. English (558,813)
2. German (228,000)
3. Japanese (116,000)
4. French (105,000)
5. Swedish (74,000)
6. Dutch (69,000)
7. Polish (66,000)
8. Spanish (48,000)
9. Portuguese (44,000)
10. Italian (42,500)
11. Chinese (27,300)
12. Danish (24,600)

Editing

en.wikipedia.org/wiki/Open_source

1 Root terminology
2 Open source model
3 Open-source license
4 Open source movement
5 Open source vs. closed source
6 Open source vs. free software
7 Participants in OSS development projects
8 Open source software development tools
    8.1 Source code revision control
    8.2 Testing tools
    8.3 Bug/Error/Defect tracking tools
    8.4 Communication
9 Influence on other fields
10 Advocates
11 Prominent projects and organizations
12 Examples of open source software
13 See also
    13.1 Related topics
    13.2 Contrast with
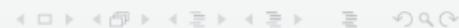14 Further reading

G. Sivakumar   Computer Science and Engineering IIT Bombay siva@iitb.ac.in
Free/Open Source Software in Engineering Curriculum

OSS Model and Tools

From the Wikipedia page examine the following.

- Open Source Model
- Open Source Participants
- OSS Development Tools
- OSS Advocates and Projects/Organizations

G. Sivakumar   Computer Science and Engineering IIT Bombay siva@iitb.ac.in
Free/Open Source Software in Engineering Curriculum

Benefits of FOSS in Academia

FOSS, like academia, is

- Parallel, rather than linear
- Large globally distributed community
- Highly talented, highly motivated collaborators.
- Truly independent *peer review*
- Prompt feedback to users/developers.
- Rapid release schedules.

Other Educational Sector Examples

- Linux Cluster (32 nodes) at IIT-Bombay
  - Most Value for money!
  - Molecular modelling
  - Circuit Simulation
  - Aerodynamics
- Online Course Management
  - dotLRN
  - Moodle
- Examples from Science and Engineering

SciLab vs Matlab

- 2-D and 3-D graphics, animation
  www.scilab.org
- Linear algebra, sparse matrices
- Polynomials and rational functions
- Simulation: ODE solver (ODEPACK) and DAE solver (DASSL)
- Scicos: a dynamic systems modeler and simulator
- Classic and robust control, LMI optimization

- Differentiable and non-diffe optimization
- Signal processing
- Metanet: graphs and netwo
- Parallel Scilab using PVM
- Statistics
- Interface with Computer Al MuPAD)

Which is better for **student?** for your *college*?

G. Sivakumar   Computer Science and Engineering IIT Bombay siva@iitb.ac.in

Free/Open Source Software in Engineering Curriculum

Civil Engineering

Finite Element Analysis

:: elCalc
A program for engineering calculation of structures (steel,concrete and other), being an add-on to any FEM (Finite Element Method) program, which is able to produce textual output.

:: femML
femML is an XML-based language for describing finite element models and associated results for inter - and intra- application data exchange and integration.

:: Meshgen
Meshgen is designed to interactively generate 2D FEM meshes composed of triangular and quadrilateral elements.

:: Pfem
Pfem is a python-based finite element program aimed at solving solid mechanics and heat transfer problems with flexibility, efficiency and sound object-oriented design.

:: Openfem
Develop finite element stress analysis programs, that work with free cad systems(Maybe QCad). Right now there's only

G. Sivakumar   Computer Science and Engineering IIT Bombay siva@iitb.ac.in

Free/Open Source Software in Engineering Curriculum

Civil Engineering

Numerical Analysis Libraries

:: GSL (GNU Scientific Library)
A modern numerical library for C and C++ programmers. The routines have been written from scratch by the GSL team in ANSI C.

:: Blitz++
Blitz++ is a C++ class library for scientific computing which provides performance on par with Fortran 77/90. It uses template techniques to achieve high performance.

:: TNT (Template Numerical Toolkit)
A collection of interfaces and reference implementations of numerical objects useful for scientific computing in C++. The toolkit defines interfaces for basic data structures, such as multidimensional arrays and sparse matrices, commonly used in numerical applications.
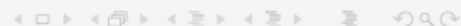
G. Sivakumar   Computer Science and Engineering IIT Bombay siva@iitb.ac.in
Free/Open Source Software in Engineering Curriculum

GRASS GIS Development

GRASS: Documentation

Home	Introduction	Download	Documents	Community

Tutorial and Courses

- English:
  - ○ GDF Hannover: GRASS 6 Course material
  - ○ GRASS 6 WIKI tutorial
  - ○ GRASS 5.0/5.3 tutorial project, especially Part III of the full tutorial: GRASS in 10 minutes - Quick
  - ○ GRASS 5.4/6 Tutorial (translation of Italian Tutorial)
  - ○ GRASS GIS: A Useful Tool for the Mountain Cartographer (Pat Dunlavey, 2002)
  - ○ An Introduction to: Geographic Resources Analysis Support System (GRASS GIS 4.x) (ISAT Envir
  - ○ Introduction to GRASS GIS software (2. ed., 1998) in English language (80 pages, online and PS)
  - ○ GRASS 4.2 Seeds Beginner's Tutorial (1995, University of Leicester/Project ASSIST)
  - ○ Spearfish database description (Postscript, 8 pages)
- Czech:
  - ○ Translation of GDF Hannover: GRASS 6 course script
  - ○ České sdružení uživatelů GISu GRASS (Jáchym Cepicky, online)
- French: French translation of the GRASS 5 tutorial project

G. Sivakumar   Computer Science and Engineering IIT Bombay siva@iitb.ac.in
Free/Open Source Software in Engineering Curriculum

GRASS GIS Development

GRASS Development

Mailing list - Download/CVS - Help wanted! - Compiling GRASS - Programmer's Manual - Code submission - Related software

GRASS - Geographic Resources Analysis Support System has been under continual development since 1982 and has involved a large numb
universities, and private companies. The core components of GRASS and the management of the integration of the efforts into GRASS relea
the Construction Engineering Resesarch Laboratory (CERL) in Champaign, Illinois. It has been estimated that several million dollars of GRASS
efforts across the government have been completed since the 4.1 release. Since 1997 a worldwide network of developers continue to releas
GRASS history).

The strength and success of GRASS relies on the user community. This in mind, the philosophy of the GRASS Development Team is to enco
own unique tools and applications for GRASS. If you develop tools and techniques that you feel would benefit other GRASS users, let us know
follow the coding style described in SUBMITTING file (which you find in the main directory of GRASS source code). GRASS GIS is developed ar
developers team [see the incomplete GRASS Developers World Map (big: 500k)]. As GRASS is growing, we have plenty of opportunities to joi
team and improve the powerful GRASS GIS under terms of GNU General Public License.

Many people have contributed to the GRASS GIS. Without any one of them, GRASS would not exist in its current form. The authors of the ind
at the end of their manual page in GRASS users manual, however, numerous authors of bug fixes and enhancements as well as people who
coordination, integration, documentation and testing are not mentioned. Therefore, this page is an attempt to acknowledge those who contr
development. Please allow us to extend our most cordial thanks to all of you. If you contributed to GRASS at any point during it's existence, p
name and e-mail address so we can add your name here.

Development happens in GRASS 6.

Developers Mailing List

Please join our mailing list which shall help and coordinate ongoing development in GRASS GIS development. Join the list if you are programr
and/or want to follow the development discussion. If you have questions concerning compilation of the CVS-GRASS, please ask here.

- **How do I subscribe to GRASS Developers Mailing List?**
  Visit the GRASS mailing list (un)subscription page

- GRASSS mailing list Archive
- Search Mailing List

How you can help

- **Bug reports:** You are kindly requested to report bugs through our bug tracking form. Our system keeps you informed about progres
- **Bug fixing:** Please assist in fixing known bugs from the GRASS known bugs list.
- **Implementation of wishes:** Open wishes are listed in the GRASS wish list.
- **Write missing man pages:** Help us to write missing man pages. Check against the existing GRASS 6 manual pages.

GRASS GIS Development

Roadmap

See the GRASS Development roadmap

Download the latest GRASS software

To follow GRASS development, get the latest code from CVS server:

- Download official GRASS releases
- CVS repository (latest code)
- New Developers: How to get GRASS CVS write access with ssh. To

Compiling GRASS

Compiling GRASS is only recommended to people with experience (that'
The **configuration** requires a set of installed libraries which are describe
code tarball). The **compilation** is desribed in the INSTALL file (which you
search the developers mailing list archive before asking developers - mo

GRASS Programmer's Manual

Please cite GRASS when using it as we have invested a lot of time and e
Citation:
GRASS Development Team (<*year*>). Geographic Resources Analysis S

- GRASS 5.0/5.3/5.4 (old):

Cost of FOSS

- There is no Free Lunch! What is the tradeoff?
- Skilled Human Resources versus Money!
- Which one does India have in **abundance?** Which one should we bet on?
- Goals of OSSRC centre (http://ossrc.org.in)
- Goals of this course (Train the Trainers)
- We need your help! (Ramakrishna sweets story)

Migrating IndiX from Xfree86 to Xorg

Siji Sunny, Soumen Debgupta and Amit Pundir

Language Computing Group, CDAC,Gulmohar Cross Road No.9, Mumbai- 4000049,India

{siji,soumen,amitp}@cdacmumbai.in

Abstract

Unix started as a CUI (Command User Interface) operating system. XWindows provided a very convenient GUI (Graphical User Interface) so that multi- tasking could be done and desktop environments could be created on the Unix platform.

Many Unix like systems exist. GNU/Linux is a user friendly, freely distributable, Unixlike system under GPL which used Xfree86 as X. IndiX had modified this Xserver to enable Indic languages on GNU/Linux. However, as there may not be further Xfree86 developments in GPL, more and more GNU/Linux distributions are switching over to Xorg (new group formed by the developers to continue Xserver developments under GPL). In this presentation, we will be discussing some of the issues that need to be sorted in order to make the versatile IndiX system work with latest Xserver (Xorg), so that the newer devices can be supported.

Introduction

The IndiX project funded by TDIL, Government of India executed at CDAC Mumbai had enabled 13 Indian Languages on GNU/Linux. In IndiX project the core windowing system of the GNU/Linux (X11 Server and Client) are enhanced with Indic capability. The X11 Server converts English characters to their shapes and in IndiX the X11 Server does the same for Indian scripts.

The basic IndiX agenda is that the text processing should be as easy for the Indian scripts as it is for English for the user. But within the computer, processing Indic text is not exactly the same as that for English.

The technological challenge addressed by IndiX is to identify the minimal, logical and required changes in Indic text processing and embed these changes within the lower most level of a widely used and deployed software architecture, GNU/Linux. Indic text processing is not a product or commodity but modifications to the X Server. Thus IndiX project Xfree86 was modified.

As Xfree86 group have discontinued further developments in GPL. In order to continue GNU Linux activities, a new group was formed by the FOSS community called as X.org and all the distributions of GNU Linux are changing their XServers from Xfree86 to Xorg.

Hence, there is a need to make IndiX technology work with Xorg.

About X

X was developed by the Athena project at MIT and was released in 1984. In 1988 an entity called the "X Consortium" took over X, and to this day it handles development and distribution. It is the standard graphical interface on Unix, Unix- like operating systems, and is available for most other modern operating systems.
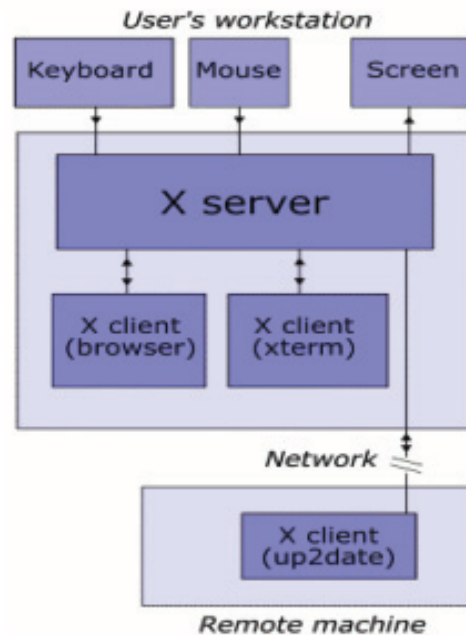
X is based on a client - server model. A client application runs on a host computer and communicates with other X servers running on

remote or local PCs. In client server architecture application themselves are clients. X servers job is to receive request from X client as drawing window from point to point. X server moreover manages different input devices (mouse, keyboard,etc.). X client builds window with various widget sets like button, menu, text box, etc. The communication protocol between server and network is transparent.

A client and server may run on the same machine or different ones, possibly with a different platform. Practically speaking, we can run a computation intensive task on a remote machine and display the result on a local computer desktop.

In most graphical interfaces there is a top level window, that is root window. The term windows is also used to denote windows that also lay within another window that is sub- windows of a parent window. Elements such as buttons, menus, icons can be realised using sub- windows. The window created by the client creates a tree architecture, root of this tree is the root window, which is a special window automatically created at server startup. All others are directly or indirectly subwindows of this root window.

IndiX

GNU/Linux was to be localized so that most applications can work almost as easily with Indic scripts as with Latin. In this project IndiX, X server was modified, so that clients can send UTF-8 encoded text in several Indian scripts like Devanagari, Kannada, Malayalam and Tamil.

What IndiX did in existing X architecture

IndiX identified three minimal and necessary enhancements to the Latin text processing [Ref:2]. The basis of these enhancements is to treat the syllable as an equivalent of a character in Latin.

1 Provide text rendering interface at the lowermost level as provided for Latin.

2 To support text editing, provide interface to get the extent of a syllable and maintain the syllable boundaries and their extents dynamically.

3 Finally, provide a virtual CharMap that will map from a syllable to the sequence of glyphs from a font. Currently, the IndiX project has enabled 13 Indian languages using this architecture.

And we are studying the Xorg architecture to port the IndiX- 2 architecture from Xfree86 to Xorg.

Merging from Xfree86 to Xorg

Xfree 86 is a freely- redistributable Open Source implementation of the X Window System by Xfree86 project - a volunteer Organization.

Xfree86 was originated in 1992 from the X386 server for IBM PC compatibles, it was included with X11R5 in 1991, written by Thomas Roell and Mark W. Snitily and donated to the MIT X Consortium by Snitily Graphics Consulting Services (SGCS).

Why Xfree86 has been rejected by FOSS community

Following were the reasons for searching an alternative for Xfree86.

1 XFree86's license is not GPL-compatible.

2 XFree86 has always allowed contributors and developers to license their work in their own names whatever manner they wish that conform to XFree86 licensing policy. This includes demanding credit for their work, which is completely against GPL.

3 The new license does not apply to client- side libraries, only the X server.

People have already contributed code to the X server having their own copyrights. If XFree86 wants to change its license under GPL, they must either contact each contributor for relicensing, or they will have to remove the code of all those who have contributed, else they would be violating the licenses of the individual contributors.

In May 1999, the Xfree86 team had joined together and formed Xorg. They were working as non- paying members, encouraged by various hardware companies interested in using Xfree86 with Linux. Now most of the Linux distributions have adopted Xorg instead of XFree86.

In the longer run, they are planning to make X modular by splitting X into a series of smaller packages for the server, different libraries and utility applications etc. Different Linux distributions want such a modular release so that different parts of X can be updated independently. Apparently, many of these components are rapidly reaching maturity - they are not rewrites. After all, just a split of the existing code into pieces with a more modern build architecture (autotools vs. imake).

Acknowledgments

We would like to thank and acknowledge Mr. Zia Saquib, Executive Director for his encouragement, Dr. Alka Irani for her guidance and our colleagues in Language computing Group CDAC Mumbai for their support.

Glossary:

(1) Syllable:

Indian language text input differs from that in English. The most significant difference of these is that in English, each keystroke maps directly to a letter. Each letter has a unique code. A *"Syllable"* - the Indian language equivalent unit of writing letter, however is composed of one or more characters entered through the keyboards or any other input mechanism. There are far too many syllables to be encoded separately. The syllable is broken down into vowels, consonants and modifiers. These are then encoded, just as roman alphabets are. The user types in a sequence of vowels, consonants, and modifiers. The machine then composes syllables at run time based on language dependent rules. Every syllable is thus represented in the machine as a unique sequence of vowels, consonants and modifiers.

(2) Unicode and UTF-8:

Developed in cooperation between the Unicode Consortium and the International Organization for Standardisation (ISO), Unicode is an attempt to consolidate the alphabets and ideographs of the world's languages into a single, international character set. It focuses on the characters themselves rather than on languages. Thus, a letter shared between English and Russian (or for that matter, an ideograph shared between kanji and Han script) would have the same Unicode character. As a multilingual standard, Unicode makes it possible for developers to create applications without having to resort to the costly, time- consuming task of releasing localized versions for each language.

UTF-8 is a compromise character encoding that can be as compact as ASCII (if the file is just plain English text) but can also contain any unicode characters (with some increase in file size). UTF stands for Unicode Transformation Format. The '8' means it uses 8- bit blocks to represent a character. The number of blocks needed to represent a character varies from 1 to 4.

(3) IndiX:

Realizing the importance of enabling open source infrastructural software like Linux to support Indic scripts, the Ministry of Information Technology, of the government of India funded the IndiX project at the CDAC Mumbai (formerly NCST) in 2000. IndiX provided the Indian Language support for the Linux Operating System. IndiX team had modified the X server so that the client applications can send Unicode based text in several Indian scripts like Devanagiri, Kannada, Malayalam and Tamil. http: //www.cdacmumbai.in/projects/indix/

(3) FOSS/GPL:

The GNU General Public License is intended to guarantee your freedom to share and change free software- - to make sure the software is free for all its users. When we speak of free software or FOSS (Free Open Source Software), we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish). Also, that you receive source code or get it if you want it, you can change the software or use pieces of it in new free programs; and that you know you can do these things.

References:

1. X window System By Robert W.Scheifler and james Gettys

2. IndiX- Indian Language support for the Linux Operating System By Vinod Kumar , Sandeep Rao , Ajit Joshi , Rekha Sharma , CDAC Mumbai. http: / /www.cdacmumbai.in/projects/ indix/

3. http:www.en.wikipedia.org

4. http: / / www.x.org

5. http: / / www.xfree86.org

Another 'world' is possible -
(through aanatarabhaaratii Multilingual portal)

Anal Haque Warsi[1], Santosh[2], Sobi[3]

1. Project Engineer-anal@cdacmumbai.in.
2. Project Engineer, -santoshg@cdacmumbai.in
3. Project Engineer , -sobi@cdacmumbai.in
Language Computing Group, C-DAC Mumbai

Abstract

Most of the Computer end users-in India are familiar with only one operating system MS-windows. The many of the end-users in India may not even be knowing that there is an alternative Operating System platform for computers. Let us group the systems under that category as Unix and Unix-like systems. GNU/Linux (sometimes referred to as Linux) falls in this second category. With Govt of India taking extra efforts to spread the computer network across the whole country to provide easy accessibility to villages, educational scene is India can change drastically if we build a good educational contents framework. In this presentation we will be discussing how using this alternative platform called GNU/Linux we can create and use educational contents in multiple languages.

Today we will be demonstrating one such service which can be used for storing, sharing and displaying multi-lingual documents in open formats like html etc... It is provided through aantarabhaaratii portal by the janabhaaratii project of CDAC Mumbai using jana-indix CD. It should be noted that all the operations can be efficiently done form the GNU/Linux platform. Thus it is very important to create awareness about GNU/Linux as a very effective platform.

This presentation is about an **Open Source cost cutting Solutions in Education using portal aantarabhaaratii, developed and maintained by janabhaaratii project of Language Computing Group at C-DAC Mumbai.**

Feature of our antrabhaaratii portal are:

●   A storehouse for Useful/relevant contents

●   Distributed content Maintenance and multiple content maintenance

●   All inclusiveness: Some Segments of the population are sometimes excluded (female, kids, minors, physically challenged, Sr. Citizens); we will create various user segments on the portal

●   Easy content development in multiple Indic languages

A brief description of aantarabhaaratii portal:

Objectives of the Portal

The portal is meant for a collaboration effort for contents in Indian Languages, initially covering 6 languages but eventually providing for all Indian Languages. The initial 6 languages are: Hindi, Marathi, Bengali, Oriya, Telugu and Malayalam. It is developed using Plone. Plone is a user-friendly powerful content managementsystem - ideal as an intranet and extranet server, as a document publishing system, a portal server and as a groupware tool for collaboration between separately located entities. **Plone** is built on top of the Python application server Zope and its accompanying Zope Content Management Framework.

Basic Features

The portal will contain the following features common to all its users:

1. Eventually, it will be developed across all the Languages of India and in all the major scripts

2. Language-wise: Currently the portal is enabled for 13 Indian languages. Gradually all the official languages will be included.

3. Dialect wise: Within each language, various dialects and regional interests will be represented with specific literature/articles of interests/ contributions.

4. Bulletin Boards: The portal will have notice boards called Bulletin boards on various issues

5. Weblogs / WIKI : The portal will have on-line

encyclopedia

6. Contents : Portal will have useful Corpa/e-books documents in multiple languages

Distributed authority structure for maintaining/ developing the Portal

Various kinds of users may be identified and these can be broadly classified as internal/external or developers/general layman or based on languages. All of the users will be from the broader 'community' and based on their expertise and level of contribution, a hierarchy of users has been identified.

To enable the hierarchical structure effectively, each user who registers with the portal gets a certain privilege level between 0(highest) and 7(lowest).

People from different organisations and different Geographical backgrounds can collaborate in developing and maintaining the portal. People can be situated at different locations and on varied platforms and yet take part in portal development.

Management of various components / products / specific portions of the portal will be delegated to the appropriate co-developers.

By sending a request to the portal administrator registering with the portal, content developers, will be part of a community that brings together project seekers and users.

They will have access to tutorials, manuals and source codes for many applications. These can be downloaded by anonymous ftp.

Developers may work for renumeration this will be the major source of turnover/revenue generation within the open source community. On the other hand, some individuals may choose to work voluntarily for causes/evangelical pursuits.

General users can register on the portal with privilege level 7 and access various services like Indian Language email, chat, address book and bulletin/discussion groups.

Users can register by providing a username and an e-mail. An initial password will be dispatched to the email . At their will, they can sign up for some or all of these services using the common username. User can use a browser (Mozilla) and give URL* http://janabhaaratii.org.in

Once in janabhaaratii, they can click on the link for Portal.

They can login using name and password they have received through e-mail.

He/she then enables the script of his/her choice to enter text.

A typical Interactions

- Registration at the portal with a username.

- Access to resources based on username privilege level.

- A user work area for contributors / co-developers.

- Downloads of software and useful resources for users.

- Posting messages on bulletin boards relevant to the user.

- Access to status pages where progress of development and translations is presented.

Our endeavor is to make technology in Indian languages reachable to the vast percentage of Indians who are not English speakers i.e. the population in Indian villages with a smooth and easy to use interface. There are various schemes available under janabhaaratii to achieve this.

Empowerment schemes under janabhaaratii

Janabhaaratii is the project to enable wide use of Indian Language computing on GNU/Linux platform. Following schemes are available through janabhaaratii:

- jana-praaMgaNa - Open day - to learn developing contents in Indian languages (Once in a month)

- jana-vyaasa-piiTha - A seminar series

- Teachers' empowerment - A free seminar for teachers once in three months.

- Each school will be given free space to develop multi-lingual contents on aantarabhaaratii portal for 2 years.

- Writers from various linguistic backgrounds will be given free space on Portal for 2 years.

- Every month - selection of the best article in each language.

Conclusion and roadmap for future:

- Increase in Computer literacy: Portal itself can be used for increasing Computer literacy as layman will find it convenient to use it for

education through entertainment.

● Content development can be facilitated.

● Teachers training can be taken up so that they can develop contents on portal.

● Awareness programs for layman can be taken up. As operator/grassroots practitioners needs to understand the basic technology, such as how to navigate the Internet or to optimize the use of the available tools

● Content editors can be selected on honorary basis: These are prestigious posts like editors of media publications. As qualitative contents needs to be maintained, indexed for searching purpose. We expect teacher community's support for taking up this job on honorarium.

● Despite having an inventive technology infrastructure, we were unable to attract users. More and more vendors are coming forward with easy to use, flexible and affordable GNU/Linux flavors.

● Awareness campaigns can be taken up by NGOs, schools, colleges, social organizations: With minimum introduction, the use of portal using alternate technology will create awareness about the alternate platform : GNU/Linux and facilitates its learning.

Some websites:

http://www.cdacmumbai.in

http://www.janabhaaratii.org.in

http://janabhaaratii.org.in:9673/aantarabhaaratii

http://www.zope.org

http://www.plone.org

"Xen Virtualization for Linux"

Amol A Sale

Fellow Department of Computer Science Univ. of Pune 411007.
Email- amoal@cs.unipune.ernet.in

Abstract

This paper provides an overview of the architecture, features, and benefits of Xen **Virtualization Technology** the most cost-effective solution for today's development industries. Virtual machine technology enables customers to run multiple operating systems concurrently on a single physical machine. Virtualization addresses a set of key customer scenarios, including consolidating and automating software test and development environments, migrating legacy applications, consolidating multiple machine workloads, and testing distributed applications on a single physical machine.

Virtualization is basically a way to run multiple operating systems at the same time on the same machine. It may be viewed as multitasking, but it is beyond that. Multitasking is mostly used when speaking of programs. Running several programs at once requires good multitasking abilities. However running multiple operating systems requires a lot more than that. The advantages of running several OSs simultaneously are huge, especially for security purposes.

Businesses continually seek ways to reduce cost and risk while increasing quality and agility in their IT infrastructure. They are always looking for new ways to help improve overall utilization and to increase the flexibility with which they can deploy their hardware to meet their changing business needs. As x86 platforms have continued to make dramatic improvements in price-to-performance value, software technologies have evolved to help businesses more effectively harness that improved performance in a manageable way. Virtualization technology is one such technology

Virtualization as of now is handled only by software. In the sense, there is no specific hardware that is dedicated for it. Intel has announced Vanderpool technology and AMD Pacifica.

Introduction to Virtualization

There are three main types of virtualization

- Para Virtualization
- Binary Translation
- Emulation

Para virtualization

Over here, the hosted OSs is made aware that they are in a virtualized environment, and are modified so will behave correctly. The OSs needs to be tweaked for this method, and there has to be back and forth between the OS coders and the virtualization coders. This is more of co-operative relationship that completes virtualization. It works very well for open source OSs where you can tweak what you want. Linux, xBSD and others are perfect PV candidates, but windows is not.

Binary Translation

Binary Translation (BT) can be looked as the middle ground. It analyses what the guest or 'hosted'OS is trying to do and changes it on the fly.

If the OS tries to execute an instruction XYZ, and XYZ will cause problems to the virtualization engine, then it will change XYZ to something more acceptable and itself generate the results of what XYZ should have returned. This has to be done quite meticulously, and can be CPU time consuming, both for the monitoring the fancy footwork required to make sure it's functioning correctly.

Emulation

It is most popular virtualization technique. The most people are familiar with it. For example, when you play Super Mario Bros on the PC, the program runs in a Super Nintendo emulator. Now suppose you have several such emulators- for instance, a play station emulator, or a virtual Atari 2600, each running a game- this can be considered as most basic form of virtualization. This is because as far as any game running in emulator is concerned, it's running on the original hardware. However emulation is really expensive in terms of processor overhead. This is because an emulator has to reproduce everything from the CPU instructions to the I/O devices in the software. It even has to provide cross CPU operation,

such as running windows software on Mac. This causes a major performance hit since it must translate every instruction, data transfer, etc.

Why virtualization

Imagine that you're a developer who need's to write a code for a program. This code will need to run on various operating systems such as windows95, 98, 2000, XP and Linux among others. Now you can either have five machines on your desk with the select OSs or one with five virtual OSs running all at once. Another relevant example would be for a web server. Suppose there are 50 users on an average computer, each running a web site, we can have 50 machines or one. Fifty is definitely the expensive way to go. Virtualization fixes this by simulating the work one by several computers. That is, users get what appears to be their own computer. For all they know, it's their own machine and no one else is using it. If they want standard SuSE or even Red Hat with custom modes that only can be understood. Through virtualization, the machine can be completely protected. Even against harmful scripts, viruses and other problems. If a code causes an error in the OS, crashing the machine, it wouldn't matter because it'd only crash that particular instance. The OSs could quickly be reloaded and you wouldn't even realize what happened.
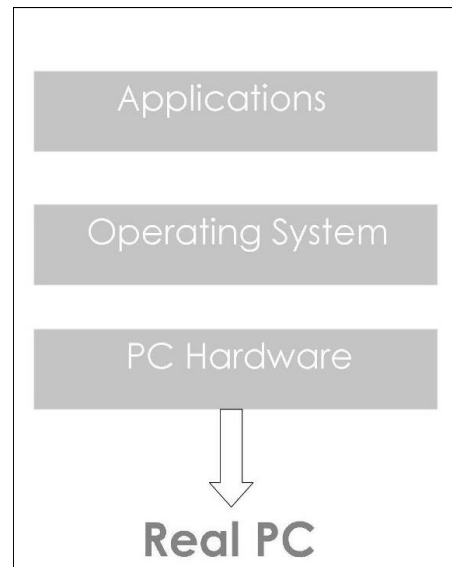
How Does It Work?

Virtualization works by setting up 'virtual machines' on the host computer. A virtual machine is a software environment that sits on top (In terms of priority and privileges) of one or more OSs and applications that actually run inside or 'under' the virtual machine. The OSs can't tell the difference between operating in 'real' machine or in 'virtual' machine. Within a virtual machine, you can do almost anything that you can do with real machine, with complete safety. A virtual machine can let you use a given platform's operating system software under another OS on the same platform, such as running a copy of Linux in a virtual machine on your windows 2000 PC. This environment is created by a Virtual Machine Monitor (VMM). The VMM can create and manage one or more virtual machines on a single real machine. Each virtual machine can provide facilities for an application or guest OS to believe it's running in a normal environment with access to physical hardware and resources.
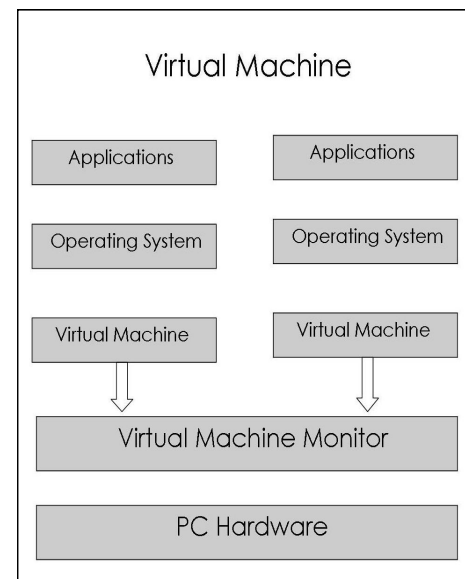
As shown in the following figure 1

a) In a real machine the OS directly communicates with the hardware.

b) The Virtual Machine Monitor (VMM) sits over the OS in terms of privileges

(a)

(b)

Fig 1: Basics of Virtual Machines

The fundamental idea behind a virtual machine is to abstract the hardware of a single computer (the CPU, memory, disk drives, network interface cards, and so forth) into several different execution environments, thereby creating the illusion that each separate execution environment is running it's own private computer.

Virtualization works by setting up 'virtual machines' on the host computer. A virtual machine

is a software environment that sits on top (In terms of priority and privileges) of one or more OSs and applications that actually run inside or 'under' the virtual machine. The OSs can't tell the difference between operating in 'real' machine or in 'virtual' machine. Within a virtual machine, you can do almost anything that you can do with real machine, with complete safety. A virtual machine can let you use a given platform's operating system software under another OS on the same platform, such as running a copy of Linux in a virtual machine on your windows 2000 PC. This environment is created by a Virtual Machine Monitor (VMM). The VMM can create and manage one or more virtual machines on a single real machine. Each virtual machine can provide facilities for an application or guest OS to believe it's running in a normal environment with access to physical hardware and resources.

By using CPU scheduling and virtual memory techniques, an operating system can create the illusion that a process has its own processor with its own (virtual) memory. Normally, a process has additional features, such as system calls and a file system that are not provided by the bare hardware. The virtual machine approach does not provide any such additional functionality but rather provides an interface that is identical to the underlying bare hardware. Each process is provided with the virtual copy of the underlying computer (fig 2).

There are several reasons for creating a virtual machine, all of which are fundamentally related to being able to share the same hardware yet run several different execution environments (i.e. different Operating Systems) concurrently.
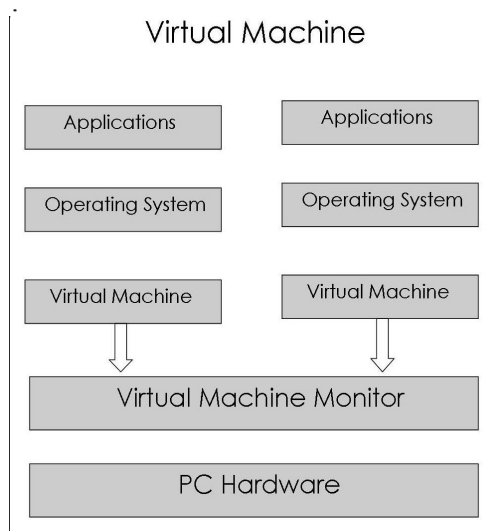

Fig 2 Virtual Machine

A major difficulty with the virtual machine approach involves disk systems. Suppose that the physical machine has three disk drives but wants to support seven virtual machines. Clearly it cannot allocate a disk drive to each virtual machine, because the virtual machine software itself will need substantial disk space to provide virtual memory and spooling. The solution is to provide virtual disks – termed minidisks in IBM's VM operating system – that are identical in every aspect except size. The system implements each minidisk by allocating as many tracks on the physical disk as the minidisk needs. Obviously the sum of the sizes of all minidisks must be smaller than the size of the physical disk space available

Users thus are given their own virtual machines. They can then run any of the operating systems or software packages that are available on the underlying machine. For the IBM VM system, a user normally runs CMS-a single user interactive operative operating system. The virtual machine software is concerned with multiprogramming multiple virtual machines onto a physical machine, but it do not need to consider any user support software. This arrangement may provide a useful way to divide the problem of designing a multi-user interactive system.

Implementation of Virtual Machines

Although the concept of virtual-machine is useful, it is difficult to implement. Much work is required to provide an exact duplicate of the underlying machine. Remember that the underlying machine has two modes: user mode and the kernel mode. The virtual machine software can run in kernel mode, since it is the operating system. The virtual machine itself can operate in only user mode. Just as the physical machine has two modes, however so must the virtual machine. Consequently, we must have a virtual user mode and a virtual kernel mode, both of which run in a physical user mode. Those actions that cause the transfer from user mode to kernel mode on a real machine (such as a system call or an attempt to execute privileged instruction) must also cause transfer from virtual user mode to virtual kernel mode on a virtual machine.

Such a transfer can be accomplished as follows. When a system call, for example, is made by a program running on a virtual machine in virtual user mode, it will cause a transfer to the virtual machine monitor in the real machine. When the virtual machine-monitor gains control, it can change the

register contents and program counter for the virtual machine to simulate the effect of system call. It can then restart the virtual machine, noting that it is now in virtual kernel mode.

Current problems faced by Software Virtual Machines

There are a number of programs available such as Xen, Microsoft Virtual PC, VMWare that will let you create and manage virtual machines. However there are a good number of setbacks that makes software virtualization a very tedious process. Let us see those problems.

For the x86 architecture (the entire PC world as we know runs on x86, except the Mac, which uses a different architecture, the PowerPC), these involves the ring based Privilege Levels. Conceptually, priority rings are a way to divide a system into privilege levels; you can have an OS running in a level that a user's program can't modify. Here if a program acts up, it won't crash the system. Instead being in a higher privilege level, the OS can take control by shutting down the offending program. Rings enforce control over various parts of the system.

There are four rings in x86. These are 0, 1, 2 and 3. The lower the number, the higher is the privilege. To understand it more simply, an application running at priority level 2 can only make changes or mess up things at level 3, and not level 1 or 0.

In practice, OSs typically runs in ring 0 while user programs run in ring 3. Virtual machines such as Microsoft virtual PC will have to run in ring 0 in order to maintain complete control. Further they will also need to keep the OS out of ring 0 in order to avoid any sort of conflicts. The solution then is to force the hosted OS to run in the lower ring, such as ring 1. This model of virtualization is called the '0/1/3' model. However there are quite a number of drawbacks to this model. OSs are used to running in ring 0 and are coded accordingly, the problem being that some instructions will only work if they are going to or from ring 0.

The other model is called as the '0/3' model. Here the virtual machine is loaded in ring 0 and the OSs and programs are all in ring 3. Everything otherwise functions just as it would on the 0/1/3 model. However the OS in ring 3 can be walked on by user programs with much greater ease, causing a lot more crashes, blue-screens and what not. But since the programs don't have to traverse rings while
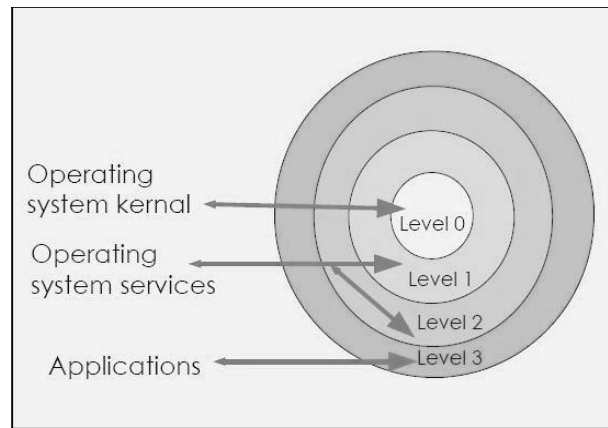

Fig 3 Protection rings

talking to the OS, everything works faster. The advantage here is speed and the drawback security.

To sum it all up, in the 0/1/3 model, you have the security at the cost of speed. In 0/3, you have only the 0 to 3 transition, so it can potentially run faster. However if a problem has to occur, it's more likely to happen in the 0/3 model than the 0/1/3 model. From a future perspective thought, 0/3 will be more widespread mainly because of 64-bit extensions to the x86 ISA do away with rings 1 and 2. So, if you want to use an x86-64 OS, you are forced to use 0/3 model.

Xen

Virtualization software vendors today charge a hefty premium (multiple of server cost) for their software, to which must be added the usual OS and application costs. But while todays virtualization products have allowed enterprises to realize significant benefits in the development, testing and QA of n-tier applications, a very high performance hypervisor is a requirement for data center wide, production-grade server consolidation and to realize the promise of a more dynamic IT infrastructure. The award winning Xen open source hypervisor, created and maintained by XenSource,is now delivering the benefits that enterprises demand from virtualization software , because it outperforms existing hypervisors by an order of magnitude while providing guaranteed service levels to each guest OS.

Moreover Xen is secure by design, runs across chipset architectures from IBM, Intel, AMD and Sun, supports all operating systems, and is freely available as open source software. Consequently Xen is being strongly endorsed by major industry players as the right way to achieve the benefits of enterprise Virtualization.

What Xen is:

Xen is the industries fastest and most secure hypervisor. Open source software, collaboratively developed by over 20 of the worlds leading enterprise infrastructure vendors. It is a common, open industry standard code base that supports all operating systems with high performance and security. It is open for innovation with the addition of additional value

Virtualization is set to become a key requirement for every server in the data center. This trend is a direct consequence of an industry wide focus on the need to reduce the Total Cost of Operation (TCO) of enterprise computing infrastructure. In spite of the widespread adoption of relatively cheap, industry standard x86-based servers, enterprises have seen costs and complexity escalate rapidly.

Today, for every dollar spent on computing hardware, as many as five dollars are spent on lifetime costs, support, maintenance, and software licenses. Operating System Virtualization, a concept pioneered by IBM in 1972 on the System 360, has become a key requirement, because it enables server consolidation, allowing multiple operating system and application images to share each server, cutting both hardware and lifetime costs.

There is another price too: vendors of virtualization software today charge a hefty premium (multiples of the server cost) for their software, to which must be added the usual OS and application costs. But while today's virtualization products have allowed enterprises to realize significant benefits in the development, testing and QA of n-tier applications, a very high performance hypervisor is a requirement for production-grade server consolidation and to realize the promise of a more dynamic IT infrastructure. Xen, a free software hypervisor, is poised to deliver these benefits, because it outperforms existing VMM by an order of magnitude while providing guaranteed service levels to each guest OS. Furthermore, Xen is freely available as free software, and is being broadly supported by major industry players.

Key benefits:

Virtualization provides the following key benefits:

Improved hardware efficiency

Virtualization provides a platform that improves hardware efficiency across a wide range of host hardware systems, can run many different x86 operating systems in the guest environment, and provides an optimized guest experience for operating systems. Policy-based management features offer both weighting and constraint methods for fine-grained control of individual virtual machines.

Increased administrator productivity

Support for virtual networking help make administrators more productive by offering scripted control of portable, connected virtual machines. These features enable easy automation of deployment and ongoing change configuration.

Consolidate and Automate Software Test and Development Environments

Virtual machine technology was developed over a quarter century ago to address these same challenges first encountered during the mainframe era, enabling side-by-side testing and production partitions on the same system.

Migrate Legacy Applications

As an IT infrastructure enables increasingly powerful and reliable solutions, a recurring challenge for many businesses is the management and maintenance of existing server-based applications. Business applications often outlive their original operating system or hardware, and as support for these primary infrastructural elements diminishes over time, cost of ownership steadily increases. Under ideal conditions, customers would prefer to continue running business applications unchanged, but three factors increase the urgency of legacy application re-hosting:

- Diminishing hardware support for legacy operating systems.

- High time and cost of administrating server-based legacy applications.

- High cost and risk to upgrade or rewrite legacy applications for increased quality and agility.

- Consolidate Workloads

Consolidation reduces the number of physical machines under management for an optimized IT infrastructure. Specific workloads demand specific approaches, each with its own benefit.

Conclusion

Virtualization represents the wave of the future for optimizing hardware utilization and datacenter

agility. Intel architecture supports flexible and cost-effective virtualization solutions today, using Xen. These solutions are already delivering substantial value in a wide range of production environments. Intel Virtualization Technology and AMD Pacifica will increase these benefits, enabling support to virtualization in an integrated and seamless fashion. By providing a new privilege layer for VMM software, and supporting key virtualization functions in hardware, Virtualization Technology will simplify VMM development and maintenance, improve interoperability with legacy OSs, enhance security and reliability, and reduce the cost and risk of implementation. Virtualization Technology is one of a series of platform advances that will be delivered over the next few years to provide critical support for enhanced datacenter flexibility, manageability, and security. Along with ongoing scaling of absolute performance and price/performance, these innovations will deliver increasing business value.

References

Book:

- **Operating System Principles :** 7th Edition Abraham Silberschatz, Peter Baer Galvin, Greg Gagne

Magazine:

- **Chip (**July 2005)

  "All OSs in one" Nikhil Hemrajani

URLs:

- http://www.cl.cam.ac.uk/research/srg/netos/papers/2003-xensosp.pdf

- http://www.cl.cam.ac.uk/research/srg/netos/xen/

- http://www.xensource.com/

- http://sourceforge.net/projects/xen/

- http://en.wikipedia.org/wiki/Xen

- http://www.intel.com

- www.amd.com

- www.ibm.com

Parallel Virtual Machine (PVM) - A Comprehensive Study

Hiren Dand[1], Sagar Kotekar-Patil[2] , Santosh Kumar Soni[3]

1. Lecturer, Mulund College of Commerce, Mumbai – 80
2. Lecturer, D. G. Ruparel College, Mumbai – 16
3. Lecturer, K. C. College, Mumbai – 20

Abstract

In this paper we describe the Parallel Virtual Machine (PVM) System and its features. PVM is a software system that permits a heterogeneous collection of computers networked together to be viewed by a user's program as a single parallel computer. PVM is the mainstay of the Heterogeneous Network Computing research project a collaborative venture between Oak Ridge National Laboratory, the University of Tennessee, Emory University and Carnegie Mellon University.

The PVM system has evolved in the past several years into a viable technology for distributed and parallel processing in a variety of disciplines. PVM supports a straight forward but functionally complete message passing model. PVM is designed to link computing resources and provide users with a parallel platform for running their computer applications irrespective of the number of different computers they use and where the computers are located. PVM is capable of harnessing the combined resources of typically heterogeneous networked computing platforms to deliver high levels of performance and functionality. In this paper we describe the architecture of the PVM system and discuss its computing model.

Introduction

Parallel processing - the method of having many small tasks solve one large problem has emerged as a key enabling technology in modern computing. The past several years have witnessed an ever-increasing acceptance and adoption of parallel processing both for high performance scientific computing and for more general-purpose application was a result of the demand for higher performance lower cost and sustained productivity. The acceptance has been facilitated by two major developments. **Massively parallel processors MPPs** and the widespread use of distributed computing. MPPs are now the most powerful computers in the world. These machines combine a few hundred to a few thousand CPUs in a single large cabinet connected to hundreds of gigabytes of memory. MPPs offer enormous computational power and are used to solve computational Grand Challenge problems such as global climate modeling and drug design. As simulations become more realistic, the computational power required to produce them grows rapidly. Thus researchers on the cutting edge turn to MPPs and parallel processing in order to get the most computational power possible. The second major development affecting scientific problem solving is distributed computing. **Distributed computing** is a process whereby a set of computers connected by a network are used collectively to solve a single large problem. As more and more organizations have high-speed local area networks interconnecting many general-purpose workstations the combined computational resources may exceed the power of a single high-performance computer. In some cases several MPPs have been combined using distributed computing to produce unequaled computational power. The most important factor in distributed computing is cost. Large MPPs typically cost more than a million. In contrast users see very little cost in running their problems on a local set of existing computers. It is uncommon for distributed computing users to realize the raw computational power of a large MPP but they are able to solve problems several times larger than they could use one of their local computers. Common between distributed computing and MPP is the notion of message passing. In all parallel processing data must be exchanged between cooperating tasks. Several paradigms have been tried including shared memory parallelizing compilers and message passing. The message passing model has become the paradigm of choice from the perspective of the number and variety of multiprocessors that support it as well as in terms of applications languages and software systems that use it. The Parallel Virtual Machine PVM system described in this paper uses the message passing model to allow programmers to exploit distributed computing across a wide variety of computer types including MPPs. A

key concept in PVM is that it makes a collection of computers appear as one large virtual machine hence its name.

Heterogeneous Network Computing

In an MPP, every processor is exactly like every other in capability, resources, software, and communication speed. Not so on a network. The computers available on a network may be made by different vendors or have different compilers. Indeed, when a programmer wishes to exploit a collection of networked computers, he may have to contend with several different types of heterogeneity:

- architecture
- data format
- computational speed
- machine load and
- network load

The set of computers available can include a wide range of architecture types such as 386/486 PC class machines, high-performance workstations, shared-memory multiprocessors, vector supercomputers, and even large MPPs. Each architecture type has its own optimal programming method. In addition, a user can be faced with a hierarchy of programming decisions. The parallel virtual machine may itself be composed of parallel computers. Even when the architectures are only serial workstations, there is still the problem of incompatible binary formats and the need to compile a parallel task on each different machine.

Data formats on different computers are often incompatible. This incompatibility is an important point in distributed computing because data sent from one computer may be unreadable on the receiving computer. Message-passing packages developed for heterogeneous environments must make sure all the computers understand the exchanged data. Unfortunately, the early message-passing systems developed for specific MPPs are not amenable to distributed computing because they do not include enough information in the message to encode or decode it for any other computer.

Even if the set of computers are all workstations with the same data format, there is still heterogeneity due to different computational speeds. As an simple example, consider the problem of running parallel tasks on a virtual machine that is composed of one supercomputer and one workstation. The programmer

must be careful that the supercomputer doesn't sit idle waiting for the next data from the workstation before continuing. The problem of computational speeds can be very subtle. The virtual machine can be composed of a set of identical workstations. But since networked computers can have several other users on them running a variety of jobs, the machine load can vary dramatically. The result is that the effective computational power across identical workstations can vary by an order of magnitude.

Like machine load, the time it takes to send a message over the network can vary depending on the network load imposed by all the other network users, who may not even be using any of the computers in the virtual machine. This sending time becomes important when a task is sitting idle waiting for a message, and it is even more important when the parallel algorithm is sensitive to message arrival time. Thus, in distributed computing, heterogeneity can appear dynamically in even simple setups.

Despite these numerous difficulties caused by heterogeneity, distributed computing offers many advantages:

- By using existing hardware, the cost of this computing can be very low.

- Performance can be optimized by assigning each individual task to the most appropriate architecture.

- One can exploit the heterogeneous nature of a computation. Heterogeneous network computing is not just a local area network connecting workstations together. For example, it provides access to different data bases or to special processors for those parts of an application that can run only on a certain platform.

- The virtual computer resources can grow in stages and take advantage of the latest computational and network technologies.

- Program development can be enhanced by using a familiar environment. Programmers can use editors, compilers, and debuggers that are available on individual machines.

- The individual computers and workstations are usually stable, and substantial expertise in their use is readily available.

- User-level or program-level fault tolerance can be implemented with little effort either in the application or in the underlying operating system.

- Distributed computing can facilitate collaborative work.

All these factors translate into reduced development and debugging time, reduced contention for resources, reduced costs, and possibly more effective implementations of an application. It is these benefits that PVM seeks to exploit. From the beginning, the PVM software package was designed to make programming for a heterogeneous collection of machines straightforward.

Trends in Distributed Computing

Stand-alone workstations delivering several tens of millions of operations per second are commonplace, and continuing increases in power are predicted. When these computer systems are interconnected by an appropriate high-speed network, their combined computational power can be applied to solve a variety of computationally intensive applications. Indeed, network computing may even provide supercomputer-level computational power. Further, under the right circumstances, the network-based approach can be effective in coupling several similar multiprocessors, resulting in a configuration that might be economically and technically difficult to achieve with supercomputer hardware.

To be effective, distributed computing requires high communication speeds.

Among the most notable advances in computer networking technology are the following:

- **Ethernet** - the name given to the popular local area packet-switched network technology invented by Xerox PARC. The Ethernet is a 10 Mbit/s broadcast bus technology with distributed access control.

- **FDDI** - the Fiber Distributed Data Interface. FDDI is a 100-Mbit/sec token-passing ring that uses optical fiber for transmission between stations and has dual counter-rotating rings to provide redundant data paths for reliability.

- **HiPPI** - the high-performance parallel interface. HiPPI is a copper-based data communications standard capable of transferring data at 800 Mbit/sec over 32 parallel lines or 1.6 Gbit/sec over 64 parallel lines. Most commercially available high-performance computers offer a HIPPI interface. It is a point-to-point channel that does not support multidrop configurations.

- **SONET** - Synchronous Optical Network. SONET is a series of optical signals that are multiples of a basic signal rate of 51.84 Mbit/sec called OC-1. The OC-3 (155.52 Mbit/sec) and OC-12 (622.08 Mbit/sec) have been designated as the customer access rates in future B-ISDN networks, and signal rates of OC-192 (9.952 Gbit/sec) are defined.

- **ATM** - Asynchronous Transfer Mode. ATM is the technique for transport, multiplexing, and switching that provides a high degree of flexibility required by B-ISDN. ATM is a connection-oriented protocol employing fixed-size packets with a 5-byte header and 48 bytes of information.

These advances in high-speed networking promise high throughput with low latency and make it possible to utilize distributed computing for years to come. Consequently, increasing numbers of universities, government and industrial laboratories, and financial firms are turning to distributed computing to solve their computational problems. The objective of PVM is to enable these institutions to use distributed computing efficiently.

PVM Overview

The PVM software provides a unified framework within which parallel programs can be developed in an efficient and straightforward manner using existing hardware. PVM enables a collection of heterogeneous computer systems to be viewed as a single parallel virtual machine. PVM transparently handles all message routing, data conversion, and task scheduling across a network of incompatible computer architectures.

The PVM computing model is simple yet very general, and accommodates a wide variety of application program structures. The programming interface is deliberately straightforward, thus permitting simple program structures to be implemented in an intuitive manner. The user writes his application as a collection of cooperating tasks. Tasks access PVM resources through a library of standard interface routines. These routines allow the initiation and termination of tasks across the network as well as communication and synchronization between tasks. The PVM message-passing primitives are oriented towards heterogeneous operation, involving strongly typed constructs for buffering and transmission. Communication constructs include those for sending and receiving data structures as well as high-level primitives such as broadcast,

barrier synchronization, and global sum.

PVM tasks may possess arbitrary control and dependency structures. In other words, at any point in the execution of a concurrent application, any task in existence may start or stop other tasks or add or delete computers from the virtual machine. Any process may communicate and/or synchronize with any other. Any specific control and dependency structure may be implemented under the PVM system by appropriate use of PVM constructs and host language control-flow statements.

Owing to its ubiquitous nature (specifically, the virtual machine concept) and also because of its simple but complete programming interface, the PVM system has gained widespread acceptance in the high-performance scientific computing community.

The PVM System

PVM is an integrated set of software tools and libraries that emulates a general-purpose, flexible, heterogeneous concurrent computing framework on interconnected computers of varied architecture. The overall objective of the PVM system is to enable such a collection of computers to be used cooperatively for concurrent or parallel computation. Detailed descriptions and discussions of the concepts, logistics, and methodologies involved in this network-based computing process are contained in the remainder of the book. Briefly, the principles upon which PVM is based include the following:

- **User-configured host pool** : The application's computational tasks execute on a set of machines that are selected by the user for a given run of the PVM program. Both single-CPU machines and hardware multiprocessors (including shared-memory and distributed-memory computers) may be part of the host pool. The host pool may be altered by adding and deleting machines during operation (an important feature for fault tolerance).

- **Translucent access to hardware**: Application programs either may view the hardware environment as an attributeless collection of virtual processing elements or may choose to exploit the capabilities of specific machines in the host pool by positioning certain computational tasks on the most appropriate computers.
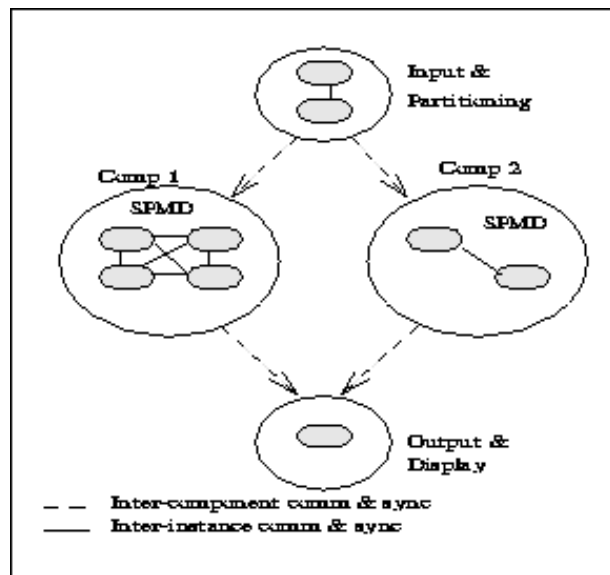
- **Process-based computation**: The unit of

parallelism in PVM is a task (often but not always a Unix process), an independent sequential thread of control that alternates between communication and computation. No process-to-processor mapping is implied or enforced by PVM; in particular, multiple tasks may execute on a single processor.

- **Explicit message-passing model**: Collections of computational tasks, each performing a part of an application's workload using data-, functional-, or hybrid decomposition, cooperate by explicitly sending and receiving messages to one another. Message size is limited only by the amount of available memory.

- **Heterogeneity support**: The PVM system supports heterogeneity in terms of machines, networks, and applications. With regard to message passing, PVM permits messages containing more than one datatype to be exchanged between machines having different data representations.

- **Multiprocessor support**: PVM uses the native message-passing facilities on multiprocessors to take advantage of the underlying hardware. Vendors often supply their own optimized PVM for their systems, which can still communicate with the public PVM version.

The PVM system is composed of two parts. The first part is a daemon , called *pvmd3* and sometimes abbreviated *pvmd* , that resides on all the computers making up the virtual machine. (An example of a daemon program is the mail program that runs in the background and handles all the incoming and outgoing electronic mail on a computer.) Pvmd3 is designed so any user with a valid login can install this daemon on a machine. When a user wishes to run a PVM application, he first creates a virtual machine by starting up PVM. (Chapter 3 details how this is done.) The PVM application can then be started from a Unix prompt on any of the hosts. Multiple users can configure overlapping virtual machines, and each user can execute several PVM applications simultaneously.

The second part of the system is a library of PVM interface routines. It contains a functionally complete repertoire of primitives that are needed for cooperation between tasks of an application. This library contains user-callable routines for message passing, spawning processes, coordinating tasks, and modifying the virtual machine.
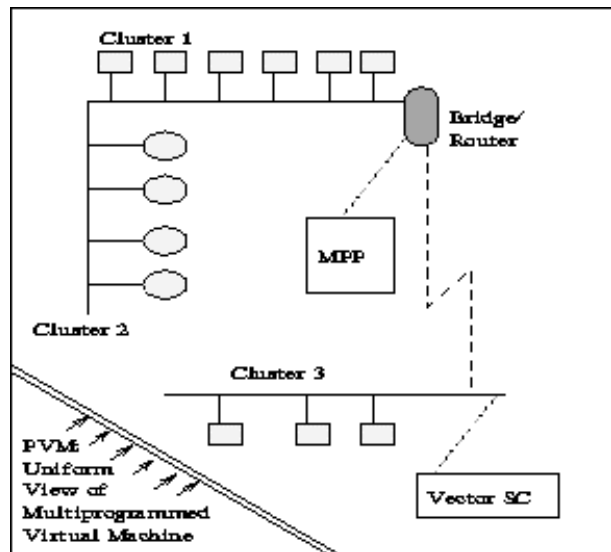
The PVM computing model is based on the notion that an application consists of several tasks. Each task is responsible for a part of the application's computational workload. Sometimes an application is parallelized along its functions; that is, each task performs a different function, for example, input, problem setup, solution, output, and display. This process is often called functional parallelism. A more common method of parallelizing an application is called data parallelism. In this method all the tasks are the same, but each one only knows and solves a small part of the data. This is also referred to as the SPMD (single-program multiple-data) model of computing. PVM supports either or a mixture of these methods. Depending on their functions, tasks may execute in parallel and may need to synchronize or exchange data, although this is not always the case. An exemplary diagram of the PVM computing model is shown in Figure. and an architectural view of the PVM system, highlighting the heterogeneity of the computing platforms supported by PVM, is shown in Figure.

(b) PVM Architectural Overview

elaborate, function arguments are a combination of value parameters and pointers as appropriate, and function result values indicate the outcome of the call. In addition, macro definitions are used for system constants, and global variables such as errno and pvm_errno are the mechanism for discriminating between multiple possible outcomes. Application programs written in C and C++ access PVM library functions by linking against an archival library (libpvm3.a) that is part of the standard distribution.

(a) PVM Computation Model

The PVM system currently supports C, C++, and Fortran languages. This set of language interfaces have been included based on the observation that the predominant majority of target applications are written in C and Fortran, with an emerging trend in experimenting with object-based languages and methodologies.

The C and C++ language bindings for the PVM user interface library are implemented as functions, following the general conventions used by most C systems, including Unix-like operating systems. To

Fortran language bindings are implemented as subroutines rather than as functions. This approach was taken because some compilers on the supported architectures would not reliably interface Fortran functions with C functions. One immediate implication of this is that an additional argument is introduced into each PVM library call for status results to be returned to the invoking program. Also, library routines for the placement and retrieval of typed data in message buffers are unified, with an additional parameter indicating the datatype. Apart from these differences (and the standard naming prefixes - *pvm_* for C, and *pvmf* for Fortran), a one-to-one correspondence exists between the two language bindings. Fortran interfaces to PVM are implemented as library stubs that in turn invoke the corresponding C routines, after casting and/or dereferencing arguments as appropriate. Thus, Fortran applications are required to link against the stubs library (libfpvm3.a) as well as the C library.

All PVM tasks are identified by an integer *task identifier* (TID). Messages are sent to and received from tids. Since tids must be unique across the entire virtual machine, they are supplied by the local pvmd

and are not user chosen. Although PVM encodes information into each TID, the user is expected to treat the tids as opaque integer identifiers. PVM contains several routines that return TID values so that the user application can identify other tasks in the system.

There are applications where it is natural to think of a *group of tasks*. And there are cases where a user would like to identify his tasks by the numbers 0 - (p - 1), where p is the number of tasks. PVM includes the concept of user named groups. When a task joins a group, it is assigned a unique "instance" number in that group. Instance numbers start at 0 and count up. In keeping with the PVM philosophy, the group functions are designed to be very general and transparent to the user. For example, any PVM task can join or leave any group at any time without having to inform any other task in the affected groups. Also, groups can overlap, and tasks can broadcast messages to groups of which they are not a member. To use any of the group functions, a program must be linked with libgpvm3.a.

The general paradigm for application programming with PVM is as follows. A user writes one or more sequential programs in C, C++, or Fortran 77 that contain embedded calls to the PVM library. Each program corresponds to a task making up the application. These programs are compiled for each architecture in the host pool, and the resulting object files are placed at a location accessible from machines in the host pool. To execute an application, a user typically starts one copy of one task (usually the "master" or "initiating" task) by hand from a machine within the host pool. This process subsequently starts other PVM tasks, eventually resulting in a collection of active tasks that then compute locally and exchange messages with each other to solve the problem. Note that while the above is a typical scenario, as many tasks as appropriate may be started manually. As mentioned earlier, tasks interact through explicit message passing, identifying each other with a system-assigned, opaque TID.

Basic Programming Techniques

Developing applications for the PVM system-in a general sense, at least-follows the traditional paradigm for programming distributed-memory multiprocessors such as the nCUBE or the Intel family of multiprocessors. The basic techniques are similar both for the logistical aspects of programming and for algorithm development. Significant

differences exist, however, in terms of (a) task management, especially issues concerning dynamic process creation, naming, and addressing; (b) initialization phases prior to actual computation; (c) granularity choices; and (d) heterogeneity. We discuss the programming process for PVM and identify factors that may impact functionality and performance.

Common Parallel Programming Paradigms

Parallel computing using a system such as PVM may be approached from three fundamental viewpoints, based on the organization of the computing tasks. Within each, different workload allocation strategies are possible and will be discussed later in this chapter. The first and most common model for PVM applications can be termed "crowd" computing : a collection of closely related processes, typically executing the same code, perform computations on different portions of the workload, usually involving the periodic exchange of intermediate results. This paradigm can be further subdivided into two categories:

- The master-slave (or host-node) model in which a separate "control" program termed the master is responsible for process spawning, initialization, collection and display of results, and perhaps timing of functions. The slave programs perform the actual computation involved; they either are allocated their workloads by the master (statically or dynamically) or perform the allocations themselves.

- The node-only model where multiple instances of a single program execute, with one process (typically the one initiated manually) taking over the noncomputational responsibilities in addition to contributing to the computation itself.

The second model supported by PVM is termed a "tree" computation. In this scenario, processes are spawned (usually dynamically as the computation progresses) in a tree-like manner, thereby establishing a tree-like, parent-child relationship (as opposed to crowd computations where a star-like relationship exists). This paradigm, although less commonly used, is an extremely natural fit to applications where the total workload is not known a priori, for example, in branch-and-bound algorithms, alpha-beta search, and recursive "divide-and-conquer" algorithms.

The third model, which we term "hybrid," can be thought of as a combination of the tree model and crowd model. Essentially, this paradigm possesses an arbitrary spawning structure: that is, at any point during application execution, the process relationship structure may resemble an arbitrary and changing graph.

We note that these three classifications are made on the basis of process relationships, though they frequently also correspond to communication topologies. Nevertheless, in all three, it is possible for any process to interact and synchronize with any other. Further, as may be expected, the choice of model is application dependent and should be selected to best match the natural structure of the parallelized program.

Crowd Computations

Crowd computations typically involve three phases. The first is the initialization of the process group; in the case of node-only computations, dissemination of group information and problem parameters, as well as workload allocation, is typically done within this phase. The second phase is computation. The third phase is collection results and display of output; during this phase, the process group is disbanded or terminated.

The master-slave model is illustrated below, using the well-known Mandelbrot set computation which is representative of the class of problems termed "embarrassingly" parallel. The computation itself involves applying a recursive function to a collection of points in the complex plane until the function values either reach a specific value or begin to diverge. Depending upon this condition, a graphical representation of each point in the plane is constructed. Essentially, since the function outcome depends only on the starting value of the point (and is independent of other points), the problem can be partitioned into completely independent portions, the algorithm applied to each, and partial results combined using simple combination schemes. However, this model permits dynamic load balancing, thereby allowing the processing elements to share the workload unevenly. In this and subsequent examples within this chapter, we only show a skeletal form of the algorithms, and also take syntactic liberties with the PVM routines in the interest of clarity. The control structure of the master-slave class of applications is shown below:
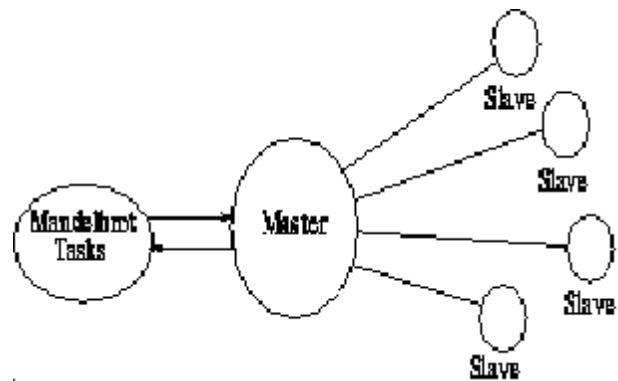
Master-slave paradigm

Tree Computations

As mentioned earlier, tree computations typically exhibit a tree-like process control structure which also conforms to the communication pattern in many instances. To illustrate this model, we consider a parallel sorting algorithm that works as follows. One process (the manually started process in PVM) possesses (inputs or generates) the list to be sorted. It then spawns a second process and sends it half the list. At this point, there are two processes each of which spawns a process and sends them one-half of their already halved lists. This continues until a tree of appropriate depth is constructed. Each process then independently sorts its portion of the list, and a merge phase follows where sorted sublists are transmitted upwards along the tree edges, with intermediate merges being done at each node. This algorithm is illustrative of a tree computation in which the workload is known in advance; a diagram depicting the process is shown below:

Workload Allocation

We discussed the common parallel programming paradigms with respect to process structure, and we outlined representative examples in the context of the PVM system. In this section we address the issue of workload allocation, subsequent to establishing process structure, and describe some common paradigms that are used in distributed-memory parallel computing. Two general methodologies are commonly used. The first, termed data decomposition or partitioning, assumes that the overall problem involves applying computational operations or transformations on one or more data structures and, further, that these data structures may be divided and operated upon. The second, called function decomposition, divides the work based on different operations or functions. In a sense, the PVM computing model supports both function

decomposition (fundamentally different tasks perform different operations) and data decomposition (identical tasks operate on different portions of the data).

Data Decomposition

As a simple example of data decomposition, consider the addition of two vectors, A[1..N] and B[1..N], to produce the result vector, C[1..N]. If we assume that P processes are working on this problem, data partitioning involves the allocation of N/P elements of each vector to each process, which computes the corresponding N/P elements of the resulting vector. This data partitioning may be done either "statically," where each process knows a priori (at least in terms of the variables N and P) its share of the workload, or "dynamically," where a control process (e.g., the master process) allocates subunits of the workload to processes as and when they become free. The principal difference between these two approaches is "scheduling." With static scheduling, individual process workloads are fixed; with dynamic scheduling, they vary as the computation progresses. In most multiprocessor environments, static scheduling is effective for problems such as the vector addition example; however, in the general PVM environment, static scheduling is not necessarily beneficial. The reason is that PVM environments based on networked clusters are susceptible to external influences; therefore, a statically scheduled, data-partitioned problem might encounter one or more processes that complete their portion of the workload much faster or much slower than the others. This situation could also arise when the machines in a PVM system are heterogeneous, possessing varying CPU speeds and different memory and other system attributes.

In a real execution of even this trivial vector addition problem, an issue that cannot be ignored is input and output. In other words, how do the processes described above receive their workloads, and what do they do with the result vectors? The answer to these questions depends on the application and the circumstances of a particular run, but in general:

1. Individual processes generate their own data internally, for example, using random numbers or statically known values. This is possible only in very special situations or for program testing purposes.

2. Individual processes independently input their data subsets from external devices. This method

is meaningful in many cases, but possible only when parallel I/O facilities are supported.

3. A controlling process sends individual data subsets to each process. This is the most common scenario, especially when parallel I/O facilities do not exist. Further, this method is also appropriate when input data subsets are derived from a previous computation within the same application.

The third method of allocating individual workloads is also consistent with dynamic scheduling in applications where interprocess interactions during computations are rare or nonexistent. However, nontrivial algorithms generally require intermediate exchanges of data values, and therefore only the initial assignment of data partitions can be accomplished by these schemes.

Function Decomposition

Parallelism in distributed-memory environments such as PVM may also be achieved by partitioning the overall workload in terms of different operations. The most obvious example of this form of decomposition is with respect to the three stages of typical program execution, namely, input, processing, and result output. In function decomposition, such an application may consist of three separate and distinct programs, each one dedicated to one of the three phases. Parallelism is obtained by concurrently executing the three programs and by establishing a "pipeline" (continuous or quantized) between them. Note, however, that in such a scenario, data parallelism may also exist within each phase.

Porting Existing Applications to PVM

In order to utilize the PVM system, applications must evolve through two stages. The first concerns development of the distributed-memory parallel version of the application algorithm(s); this phase is common to the PVM system as well as to other distributed-memory multiprocessors. The actual parallelization decisions fall into two major categories - those related to structure, and those related to efficiency. For structural decisions in parallelizing applications, the major decisions to be made include the choice of model to be used (i.e., crowd computation vs. tree computation and data decomposition vs. function decomposition). Decisions with respect to efficiency when parallelizing for distributed-memory environments are generally oriented toward minimizing the frequency and volume of communications. It is

typically in this latter respect that the parallelization process differs for PVM and hardware multiprocessors; for PVM environments based on networks, large granularity generally leads to better performance. With this qualification, the parallelization process is very similar for PVM and for other distributed-memory environments, including hardware multiprocessors.

The parallelization of applications may be done ab initio, from existing sequential versions, or from existing parallel versions. In the first two cases, the stages involved are to select an appropriate algorithm for each of the subtasks in the application, usually from published descriptions or by inventing a parallel algorithm, and to then code these algorithms in the language of choice (C, C++, or Fortran 77 for PVM) and interface them with each other as well as with process management and other constructs. Parallelization from existing sequential programs also follows certain general guidelines, primary among which are to decompose loops, beginning with outermost loops and working inward. In this process, the main concern is to detect dependencies and to partition loops such that the dependencies are preserved while allowing for concurrency. This parallelization process is described in numerous textbooks and papers on parallel computing, although few textbooks discuss the practical and specific aspects of transforming a sequential program to a parallel one.

Existing parallel programs may be based upon either the shared-memory or distributed-memory paradigms. Converting existing shared-memory programs to PVM is similar to converting from sequential code, when the shared-memory versions are based upon vector or loop-level parallelism. In the case of explicit shared memory programs, the primary task is to locate synchronization points and replace these with message passing. In order to convert existing distributed-memory parallel code to PVM, the main task is to convert from one set of concurrency constructs to another. Typically, existing distributed memory parallel programs are written either for hardware multiprocessors or other networked environments such as p4 or Express. In both cases, the major changes required are with regard to process management. For example, in the Intel family of DMMPs, it is common for processes to be started from an interactive shell command line. Such a paradigm should be replaced for PVM by either a master program or a node program that takes responsibility for process spawning. With regard to

interaction, there is, fortunately, a great deal of commonality between the message-passing calls in various programming environments. The major differences between PVM and other systems in this context are with regard to (a) process management and process addressing schemes; (b) virtual machine configuration/reconfiguration and its impact on executing applications; (c) heterogeneity in messages as well as the aspect of heterogeneity that deals with different architectures and data representations; and (d) certain unique and specialized features such as signaling, and task scheduling methods.

How PVM Works

In this part we describe the implementation of the PVM software and the reasons behind the basic design decisions. The most important goals for PVM 3 are fault tolerance, scalability, heterogeneity, and portability. PVM is able to withstand host and network failures. It doesn't automatically recover an application after a crash, but it does provide polling and notification primitives to allow fault-tolerant applications to be built. The virtual machine is dynamically reconfigurable. This property goes hand in hand with fault tolerance: an application may need to acquire more resources in order to continue running once a host has failed. Management is as decentralized and localized as possible, so virtual machines should be able to scale to hundreds of hosts and run thousands of tasks. PVM can connect computers of different types in a single session. It runs with minimal modification on any flavor of Unix or an operating system with comparable facilities (multitasking, networkable). The programming interface is simple but complete, and any user can install the package without special privileges.

To allow PVM to be highly portable, we avoid the use of operating system and language features that would be hard to retrofit if unavailable, such as multithreaded processes and asynchronous I/O. These exist in many versions of Unix, but they vary enough from product to product that different versions of PVM might need to be maintained. The generic port is kept as simple as possible, though PVM can always be optimized for any particular machine.

We assume that *sockets* are used for interprocess communication and that each host in a virtual machine group can connect directly to every other host via TCP and UDP protocols. The requirement of full IP connectivity could be removed by specifying message routes and using the pvmds to forward messages. Some multiprocessor machines

don't make sockets available on the processing nodes, but do have them on the front-end (where the pvmd runs).

XPVM

It is often useful and always reassuring to be able to see the present configuration of the virtual machine and the status of the hosts. It would be even more useful if the user could also see what his program is doing-what tasks are running, where messages are being sent, etc. The PVM GUI called XPVM was developed to display this information and more.

XPVM combines the capabilities of the PVM console, a performance monitor, and a call-level debugger into a single, easy-to-use X-Windows interface. XPVM is available from netlib in the directory pvm3/xpvm. It is distributed as precompiled, ready-to-run executables for SUN4, RS6K, ALPHA, SUN4SOL2, HPPA, and SGI5. The XPVM source is also available for compiling on other machines.

XPVM is written entirely in C using the TCL/TK toolkit and runs just like another PVM task. If a user wishes to build XPVM from the source, he must first obtain and install the TCL/TK software on his system. TCL and TK were developed by John Ousterhout at Berkeley and can be obtained by anonymous ftp to sprite.berkeley.edu The TCL and XPVM source distributions each contain a README file that describes the most up-to-date installation procedure for each package respectively.

References

1. PVM: Parallel Virtual Machine **A Users' Guide and Tutorial for Networked Parallel Computing** by Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek and Vaidy Sunderam

2. http://mjfrazer.org/~mjfrazer/uw/pvm/

3. http://www.netlib.org/pvm3/book/

MPI: A Comprehensive Study

Sagar Kotekar-Patil[1] , Hiren Dand[2] , Santosh Kumar Soni[3]

1. Lecturer, D. G. Ruparel College, Mumbai – 16
2. Lecturer, Mulund College of Commerce, Mumbai – 80
3. Lecturer, K. C. College, Mumbai – 20

Abstract

In this paper we describe Message Passing Interface (MPI) System and its features. The Message Passing Interface (MPI) is a <u>computer</u> communications <u>protocol</u>. It is a de facto standard for communication among the nodes running a parallel program on a distributed memory system. MPI implementations consist of a library of routines that can be called from Fortran, C, C++ and Ada programs. The advantage of MPI over older message passing libraries is that it is both portable (because MPI has been implemented for almost every distributed memory architecture) and fast (because each implementation is optimized for the <u>hardware</u> on which it runs). Also we have described some new implementations of MPI (getting inspired by PVM) inheriting some features from PVM like fault tolerance, language interoperabality etc.

Overview and Goals

Message passing is a paradigm used widely on certain classes of parallel machines, especially those with distributed memory. Although there are many variations, the basic concept of processes communicating through messages is well understood. Over the last ten years, substantial progress has been made in casting significant applications in this paradigm. Each vendor has implemented its own variant. More recently, several systems have demonstrated that a message passing system can be efficiently and portably implemented. It is thus an appropriate time to try to define both the syntax and semantics of a core of library routines that will be useful to a wide range of users and efficiently implementable on a wide range of computers.

A preliminary draft proposal, known as MPI1, was put forward by Dongarra, Hempel, Hey, and Walker in November 1992, and a revised version was completed in February 1993. MPI1 embodied the main features that were identified at the Williamsburg workshop as being necessary in a message passing standard. Since MPI1 was primarily intended to promote discussion and "get the ball rolling", it focused mainly on point-to-point communications. MPI1 brought to the forefront a number of important standardization issues, but did not include any collective communication routines and was not thread-safe.

In November 1992, a meeting of the MPI working group was held in Minneapolis, at which it was decided to place the standardization process on a more formal footing, and to generally adopt the procedures and organization of the High Performance FORTRAN Forum. Subcommittees were formed for the major component areas of the standard, and an email discussion service established for each. In addition, the goal of producing a draft MPI standard by the Fall of 1993 was set. To achieve this goal the MPI working group met every 6 weeks for two days throughout the first 9 months of 1993, and presented the draft MPI standard at the Supercomputing 93 conference in November 1993. These meetings and the email discussion together constituted the MPI Forum, membership of which has been open to all members of the high performance computing community.

The main advantages of establishing a message-passing standard are portability and ease-of-use. In a distributed memory communication environment in which the higher level routines and/or abstractions are build upon lower level message passing routines the benefits of standardization are particularly apparent. Furthermore, the definition of a message passing standard, such as that proposed here, provides vendors with a clearly defined base set of routines that they can implement efficiently, or in some cases provide hardware support for, thereby enhancing scalability.

The goal of the Message Passing Interface simply stated is to develop a widely used standard for writing message-passing programs. As such the interface should establish a practical, portable, efficient, and flexible standard for message passing.

A complete list of goals follows.

The first task of the MPI Forum was to define the goals that would guide its subsequent discussions. Some of these goals (and some of their implications) were the following.

- MPI would be a library for writing application programs, not a distributed operating system. This goal has implications for resource management issues.

- MPI would not mandate thread-safe implementations, but its specification would allow them. Thread safety implies that there can be no notion of a "current" buffer, message, error code, etc. As the "nodes" in the network become symmetric multiprocessors, thread safety becomes increasingly important in a heterogeneous, networked environment.

- MPI would be capable of delivering high performance on high-performance systems. Hence, no memory copies would be mandated by the design. Scalability, combined with correctness, for collective operations required that groups be "static".

- MPI would be modular, to accelerate the development of portable parallel libraries. Modularity has many implications. For example, all references must be relative to a module, not the entire program. Hence, process source/destination must be specified by rank in a group rather than by an absolute identifier and context must not be a visible value. There are many others, some of which are described below.

- MPI would be extensible to meet future needs and developments. This led to an object-oriented style without a commitment to an object-oriented language. This approach required functions to manipulate the objects, which is one minor reason for the relatively large number of functions in MPI.

- MPI would support heterogeneous computing the (MPI Datatype object allows implementations to be heterogeneous) although it would not require that all implementations be heterogeneous.

- MPI would require well-defied behavior no race conditions or avoidable implementation-specific behavior.

- Design an application programming interface (not necessarily for compilers or a system implementation library).

- Allow efficient communication: Avoid memory-to-memory copying and allow overlap of computation and communication and offload to communication co-processor, where available.

- Allow for implementations that can be used in a heterogeneous environment.

- Allow convenient C and Fortran 77 bindings for the interface. Semantics of the interface should be language independent.

- Assume a reliable communication interface: the user need not cope with communication failures. Such failures are dealt with by the underlying communication subsystem.

- Define an interface that is not too different from current practice, such as PVM, NX, Express, p4, etc., and provides extensions that allow greater flexibility.

- Define an interface that can be implemented on many vendor's platforms, with no significant changes in the underlying communication and system software.

- The interface should be designed to allow for thread-safety.

Finally, the MPI Forum sought to simplify the interface by making each approach solve as many problems as possible. For example, datatypes solve both heterogeneity and noncontiguous data layouts, both for messages and for files.

Who Should Use This Standard?

This standard is intended for use by all those who want to write portable message-passing programs in Fortran 77 and C. This includes individual application programmers, developers of software designed to run on parallel machines, and creators of environments and tools. In order to be attractive to this wide audience, the standard must provide a simple, easy-to-use interface for the basic user while not semantically precluding the high-performance message-passing operations available on advanced machines.

What Platforms Are Targets For Implementation?

The attractiveness of the message-passing paradigm at least partially stems from its wide portability. Programs expressed this way may run on distributed-memory multiprocessors, networks of workstations, and combinations of all of these. In addition, shared-memory implementations are possible. The paradigm will not be made obsolete by architectures combining the shared and distributed-memory views, or by increases in network speeds. It thus should be both possible and useful to implement this standard on a great variety of machines, including those "machines" consisting of collections of other machines, parallel or not, connected by a communication network.

The interface is suitable for use by fully general MIMD programs, as well as those written in the more restricted style of SPMD. Although no explicit support for threads is provided, the interface has been designed so as not to prejudice their use. With this version of MPI no support is provided for dynamic spawning of tasks.

MPI provides many features intended to improve performance on scalable parallel computers with specialized interprocessor communication hardware. Thus, we expect that native, high-performance implementations of MPI will be provided on such machines. At the same time, implementations of MPI on top of standard Unix interprocessor communication protocols will provide portability to workstation clusters and heterogenous networks of workstations. Several proprietary, native implementations of MPI, and a public domain, portable implementation of MPI are in progress at the time of this writing

What Is Included In The Standard?

The standard includes:

- Point-to-point communication
- Collective operations
- Process groups
- Communication contexts
- Process topologies
- Bindings for Fortran 77 and C
- Environmental Management and inquiry
- Profiling interface

What Is Not Included In The Standard?

The standard does not specify:

- Explicit shared-memory operations
- Operations that require more operating system support than is currently standard; for example, interrupt-driven receives, remote execution, or active messages
- Program construction tools
- Debugging facilities
- Explicit support for threads
- Support for task management
- I/O functions

There are many features that have been considered and not included in this standard. This happened for a number of reasons, one of which is the time constraint that was self-imposed in finishing the standard. Features that are not included can always be offered as extensions by specific implementations. Perhaps future versions of MPI will address some of these issues.

Fault Tolerance

There are several important issues to consider when providing a fault notification scheme. For example, a task might request notification of an event after it has already occurred. PVM immediately generates a notify message in response to any such after-the-fact request. For example, if a "task exit" notification request is posted for a task that has already exited, a notify message is immediately returned. Similarly, if a "host exit", request is posted for a host that is no longer part of the virtual machine, a notify message is immediately returned. It is possible for a "host add" notification request to occur simultaneously with the addition of a new host into the virtual machine. To alleviate this race condition, the user must poll the virtual machine after the notify request to obtain the complete virtual machine configuration. Subsequently, PVM can then reliably deliver any new "host add" notifies.

The current MPI standard does not include any mechanisms for fault tolerance, although the upcoming MPI-2 standard will include a notify scheme similar to PVM's. The problem with the MPI-1 model in terms of fault tolerance is that the tasks and hosts are considered to be static. An MPI-1 application must be started as a single group of executing tasks. If a task or computing resource should fail, the entire MPI-1 application must fail.

This is certainly effective in terms of preventing leftover or hung tasks. However, there is no way for an MPI program to gracefully handle a fault.

Why does MPI not guarantee buffering?

MPI does not guarantee to buffer arbitrary messages because memory is a finite resource on all computers. Thus, all computers will fail under sufficiently adverse communication loads. Different computers at different times are capable of providing differing amounts of buffering, so if a program relies on buffering it may fail under certain conditions, but work correctly under other conditions. This is clearly undesirable.

Given that no message passing system can guarantee that messages will be buffered as required under all circumstances, it might be asked why MPI does not guarantee a minimum amount of memory available for buffering. One major problem is that it is not obvious how to specify the amount of buffer space that is available, nor is it easy to estimate how much buffer space is consumed by a particular program.

Different buffering policies make sense in different environments. Messages can be buffered at the sending node or at the receiving node, or both. In the former case,

- buffers can be dedicated to one destination in one communication domain,

- or dedicated to one destination for all communication domains,

- or shared by all outgoing communications,

- or shared by all processes running at a processor node,

- or part of the buffer pool may be dedicated, and part shared.

Similar choices occur if messages are buffered at the destination. Communication buffers may be fixed in size, or they may be allocated dynamically out of the heap, in competition with the application. The buffer allocation policy may depend on the size of the messages (preferably buffering short messages), and may depend on communication history (preferably buffering on busy channels).

The choice of the right policy is strongly dependent on the hardware and software environment. For instance, in a dedicated environment, a processor with a process blocked on

a send is idle and so computing resources are not wasted if this processor copies the outgoing message to a buffer. In a time shared environment, the computing resources may be used by another process. In a system where buffer space can be in paged memory, such space can be allocated from heap. If the buffer space cannot be paged, or has to be in kernel space, then a separate buffer is needed. Flow control may require that some amount of buffer space be dedicated to each pair of communicating processes.

The optimal strategy strongly depends on various performance parameters of the system: the bandwidth, the communication start-up time, scheduling and context switching overheads, the amount of potential overlap between communication and computation, etc. The choice of a buffering and scheduling policy may not be entirely under the control of the MPI implementor, as it is partially determined by the properties of the underlying communication layer. Also, experience in this arena is quite limited, and underlying technology can be expected to change rapidly: fast, user-space interprocessor communication mechanisms are an active research area.

Attempts by the MPI Forum to design mechanisms for querying or setting the amount of buffer space available to standard communication led to the conclusion that such mechanisms will either restrict allowed implementations unacceptably, or provide bounds that will be extremely pessimistic on most implementations in most cases. Another problem is that parameters such as buffer sizes work against portability. Rather then restricting the implementation strategies for standard communication, the choice was taken to provide additional communication modes for those users that do not want to trust the implementation to make the right choice for them.

Heterogeneous Computing with MPI

Heterogeneous computing uses different computers connected by a network to solve a problem in parallel. With heterogeneous computing a number of issues arise that are not applicable when using a homogeneous parallel computer. For example, the computers may be of differing computational power, so care must be taken to distribute the work between them to avoid load imbalance. Other problems may arise because of the different behavior of floating point arithmetic on different machines. However, the two most fundamental issues that must be faced in

heterogeneous computing are,

- incompatible data representation,
- interoperability of differing implementations of the message passing layer.

Incompatible data representations arise when computers use different binary representations for the same number. In MPI all communication routines have a datatype argument so implementations can use this information to perform the appropriate representation conversion when communicating data between computers.

Interoperability refers to the ability of different implementations of a given piece of software to work together as if they were a single homogeneous implementation. A interoperability prerequisite of interoperability for MPI would be the standardization of the MPI's internal data structures, of the communication protocols, of the initialization, termination and error handling procedures, of the implementation of collective operations, and so on. Since this has not been done, there is no support for interoperability in MPI. In general, hardware-specific implementations of MPI will not be interoperable. However it is still possible for different architectures to work together if they both use the same portable MPI implementation.

References

1. Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, Jack Dongarra, MPI: The Complete Reference.

2. William Gropp and Ewing Lusk. Fault Tolerance in MPI Programs.

3. G. A. Geist, J. A. Kohl, P. M. Papadopoulos. PVM and MPI: A Comparison of Features. May 30, 1996.

4. William Gropp and Ewing Lusk. PVM and MPI Are Completely Different.

PVM and MPI : A Comparison

Santosh Kumar Soni[1], Sagar Kotekar-Patil[2] , Hiren Dand[3]

1. Lecturer, K. C. College, Mumbai – 20
2. Lecturer, D. G. Ruparel College, Mumbai – 16
3. Lecturer, Mulund College of Commerce, Mumbai – 80

Abstract:

This paper compares PVM and MPI, two systems for programming clusters, pointing out the differences in their origins, the goals of the two systems, and the relationship in terms of similarities and dissimilarities between their specifications and their implementations. It also points out the situations where one API may be favored over the other, enabling programmers to assess the needs of their applications and decide accordingly.

PVM is better when applications will be run over heterogeneous networks. It has good interoperability between different hosts. PVM allows the development of fault tolerant applications that can survive host or task failures. Because the PVM model is built around the virtual machine concept (not present in the MPI model), it provides a powerful set of dynamic resource manager and process control functions.

MPI is expected to be faster within a large multiprocessor. It has many more point-to-point and collective communication options than PVM. This can be important if an algorithm is dependent on the existence of a special communication option. MPI also has the ability to specify a logical communication topology.

Each API has its unique strengths and this will remain so into the foreseeable future. One area of future research is to come up with a hybrid model, a programming environment that allows access to the best features of both, i.e., virtual machine features of PVM and the message passing features of MPI.

Introduction

The recent emergence of the MPI (Message Passing Interface) specification has caused many programmers to wonder whether they should write their applications in MPI or use PVM (Parallel Virtual Machine). PVM is the existing de facto standard for distributed computing and MPI is being touted as the future message passing standard.   A related concern of users is whether they should invest the time and effort to rewrite their existing PVM applications in MPI.

In this paper we address these questions by comparing the features supplied by PVM and the features supplied by MPI and showing under which situations one API might be favored over another. Programmers can then assess the needs of their application and decide accordingly. Also, in this paper we have focused on a few of the many similarities and dissimilarities between MPI and PVM. We have shown that the differences between MPI and PVM remain profound, despite some convergence.

Background

Some background material will help better illustrate PVM and MPI –

What is PVM?

The impetus for developing PVM was that developers needed a framework to explore the Heterogeneous Distributed Computing and so developed the concept of a Parallel Virtual Machine.

■ PVM (Parallel Virtual Machine) is a portable message-passing system. It is designed to link separate host machines to form a "virtual machine" which is a single, manageable computing resource. This virtual machine is portable to a wide variety of architectures, including PCs, workstations, multiprocessors, and supercomputers.

In terms of background, PVM is a by production of an ongoing heterogeneous network computing research project. The general goals of this project are to develop solutions for the heterogeneous concurrent computing and to satisfy the demand for higher performance, lower cost, and sustained productivity.

■ PVM Overview – PVM transparently handles all message routing, data conversion, and task scheduling across a network of incompatible computer architectures. The programming interface is deliberately straightforward:

- The user writes an application as a collection of cooperating tasks.

- Tasks access PVM resources through a library of standard interface routines.

- These routines allow the initiation and termination of tasks across the network as well as communication and synchronization between tasks.

Owing to its virtual machine concept and its simple but complete programming interface, the PVM system has gained widespread acceptance in the high performance scientific computing community.

- PVM supported Architectures/ OSs / Languages –

- Architectures/OSs – PVM software is very portable. It has been used on all the following systems. The virtual machine can be composed of a mixture of any of these computers -

Pentium, Duals and Quads	Win95, 98, NT 4.0, Linux, Solaris, SCO, NetBSD, FreeBSD
MAC	NetBSD
Amiga	NetBSD

- Workstations and Shared-memory Servers -

SUN3, SUN4, SPARC Ultra SPARC	Sun OS, Solaris 2.x
IBM RS6000, J30	AIX 3.x, AIX 4.x
HP9000	Hpux
DEC Alpha, Pmax, Microvax	OSF, NT – Alpha
SGI	IR IS 5.x, IR IS 6.x

- Parallel Computers -

Cray YMP, T3D, T3E, Cray 2; IBM SP2, 3090; NEC SX-3; TMC CM5; Intel Paragon; Amdahl; Convex Exemplar.

- Languages -

The PVM system currently supports C, C++, and Fortran languages. Its programming library (libpvm3.a) is written in C and directly supports C and C++ applications. And, about the Fortran library (libfpvm3.a), is a set of "wrapper" functions that conform to the Fortran calling conventions.

What is MPI?

The impetus for developing MPI was that each Massively Parallel Processor (MPP) vendor was creating their own message-passing API.

MPI stands for Message Passing Interface. The goal of MPI, simply stated, is to develop a widely used standard for writing message – passing programs. As such the interface attempts to establish a practical, portable, efficient, and flexible standard for message passing.

In designing MPI the MPI Forum sought to make use of the most attractive features of a number of existing message passing systems, rather than selecting one of them and adopting it as the standard. Thus, MPI has been strongly influenced by work at the IBM T. J. Watson Research Center, Intel's NX/2, Express, nCUBE's Vertex, p4, and PARMACS. Other important contributions have come from Zipcode, Chimp, PVM, Chameleon, and PICL.

The main advantages of establishing a message-passing standard are portability and ease-of-use. In a distributed memory communication environment in which the higher level routines and/or abstractions are build upon lower level message passing routines the benefits of standardization are particularly apparent. Furthermore, the definition of a message passing standard provides vendors with a clearly defined base set of routines that they can implement efficiently, or in some cases provide hardware support for, thereby enhancing scalability.

Therefore, it is not difficult for us to know that there will exist different features between them (PVM and MPI) because of their different design issues.

Features Comparison -

This paper compares PVM and MPI with respect to the following features –

Features	PVM	MPI
Portability	Yes	Yes
Interoperability	Yes	No
Fault Tolerance	Yes	No
Process Control	Yes	MPI-2: Yes
Resource Control	Dynamic	Static
Message – Passing Topology	No	Yes
Message Handlers	Yes	Yes

What is not different (similarities) –

Despite their differences, PVM and MPI certainly have features in common. In this section, we review some of the similarities and, in the process, correct some common misconceptions about the MPI specification. In most cases these misconceptions arise because of confusion between specification and implementation, which will be covering in the subsequent section.

■ Portability - Both PVM and MPI are portable; the specification of each is machine independent, and implementations are available for a wide variety of machines, particularly those likely to appear in clusters.

Both PVM and MPI had portability as an original goal. As we have seen, MPI's very strict adherence to this principle prevented it from having some features desirable on workstation networks precisely because they could not be implemented in all environments. PVM, defined primarily by a single implementation for workstation networks, has more freedom to add features appropriate for that environment, but at the cost of making some PVM programs not portable to more restrictive environments.

■ Heterogeneity – Once a system is portable, the issue of homogeneity can be addressed. Can two processes on different machine architectures communicate with one another despite differences in byte ordering in memory or even word length? To this end PVM provides the pvm pack/unpack functions and the datatype arguments to pvm send/recv; MPI does the same with its more general MPI Datatype argument to many routines. Of course, some implementations of MPI, particularly those from hardware vendors, may not be used in a heterogeneous environment, but the MPI specification is designed to encourage heterogeneous implementations, and both the MPICH and LAM implementations support heterogeneous environments.

Both MPI and PVM permit different processes of a parallel program to execute different executable binary files. (This would be required in a heterogeneous implementation, in any case.) That is, both PVM and MPI support MIMD programs as well as SPMD programs, although again some implementations may not do so, and launching MIMD programs may be less convenient than

launching SPMD programs. Both MPICH and LAM support MIMD programming.

■ Process Control – Process control refers to the ability to start and stop tasks, to find out which tasks are running, and possibly where they are running. PVM contains all of these capabilities. In contrast MPI-1 has no defined method to start a parallel application, but MPI-2 will contain functions to start a group of tasks and to send a kill signal to a group of tasks (and possibly other signals as well).

■ Message Handler – User-level message handlers provide an extensible mechanism for building event-driven codes that easily co-exist with traditional messaging. Both PVM 3.4 and MPI-2 will have user-level message handlers. A program may register a handler function so that when a specified message arrives at a task, the function is executed.

What is different (dissimilarities) –

Major feature differences between MPI and PVM –

I. Interoperability – This term refers to the possibility of communicating among processes linked with two completely different implementations.

Heterogeneity is becoming increasingly important for high performance computing. Massively parallel processors appear to be a dying breed, leading scientists with serious computational needs to look towards clusters of smaller multiprocessors connected by new high-speed networks. Many organizations already use a variety of different computing systems in the form of different personal computers or workstations on their employees' desks. Integrating these desktop machines and utilizing their unused cycles can be an effective way of obtaining reasonable computational power. Parallel software systems therefore need to accommodate execution on many different vendor platforms.

The MPI interface was developed with the intent of encompassing all of the message-passing constructs and features of various MPP and networked clusters so that programs would execute on each type of system. The portability achieved by MPI means that a program written for one architecture can be copied to a second architecture, compiled and executed without modification.

PVM also supports this level of portability, but expands the definition of portable to include

interoperable. PVM programs similarly can be copied to different architectures, compiled and executed without modification. However, the resulting PVM executables can also communicate with each other. In other words, an MPI application can run, as a whole, on any single architecture and is portable in that sense. But a PVM program can be ported heterogeneously to run cooperatively across any set of different architectures at the same time (i.e., interoperate). While the MPI standard does not prohibit such heterogeneous cooperation, it does not require it. Nothing in the MPI standard describes cooperation across heterogeneous networks and architectures. And there is no impetus for one vendor to make its MPI implementation slower in order to allow a user to use another vendor's machine. None of the existing MPI implementations can interoperate.

PVM and MPI also differ in their approach to language interoperability. In PVM, a C program can send a message that is received by a Fortran program and vice-versa. In contrast, a program written in C is not required by the MPI standard to communicate with a program written in Fortran, even if executing on the same architecture. This restriction occurs because C and Fortran support fundamentally different language interfaces, causing difficulty in defining a consistent standard interface that covers both. The MPI decision was to not force the two languages to interoperate.

II. Virtual Machine - PVM is built around the concept of a virtual machine which is a dynamic collection of (potentially heterogeneous) computational resources managed as a single parallel computer. The virtual machine concept is fundamental to the PVM perspective and provides the basis for heterogeneity, portability, and encapsulation of functions that constitute PVM.

It is the virtual machine concept that has revolutionized heterogeneous distributed computing by linking together different workstations, personal computers and massively parallel computers to form a single integrated computational engine. In contrast, MPI has focused on message-passing and explicitly states that resource management and the concept of a virtual machine are outside the scope of the MPI (1 and 2) standard.

a. Process Control – As stated earlier, MPI-1 has no defined method to start a parallel application (whereas, MPI-2 will support).

b. Resource Control – In terms of resource management, PVM is inherently dynamic in nature.

Computing resources, or hosts, can be added or deleted either from a system console or even from within the user's application Allowing applications to interact with and manipulate their computing environment provides a powerful paradigm for load balancing, task migration, and fault tolerance. The virtual machine provides a framework for determining which tasks are running and supports naming services so that independently spawned tasks can find each other and cooperate.

Another aspect of virtual machine dynamics relates to efficiency. User applications can exhibit potentially changing computational needs over the course of their execution. Hence, a message-passing infrastructure should provide flexible control over the amount of computational power being utilized. For example, consider a typical application which begins and ends with primarily serial computations, but contains several phases of heavy parallel computation. A large MPP need not be wasted as part of the virtual machine for the serial portions, and can be added just for those portions when it is of most value. Likewise, consider a long-running application in which the user occasionally wishes to attach a graphical front-end to view the computation's progress. Without virtual machine dynamics, the graphical workstation would have to be allocated during the entire computation   . MPI lacks such dynamics and is, in fact, specifically designed to be static in nature to improve performance. There is clearly a trade-off in flexibility and efficiency for this extra margin of performance.

c. Message – Passing Topology – Although MPI does not have a concept of a virtual machine. MPI does provide a higher level of abstraction on top of the computing resources in terms of the message-passing topology. In MPI a group of tasks can be arranged in a specific logical interconnection topology. Communication among tasks then takes place within that topology with the hope that the underlying physical network topology will correspond and expedite the message transfers. PVM does not support such an abstraction, leaving the programmer to manually arrange tasks into groups with the desired communication organization.

III. Fault Tolerance – There are several important issues to consider when providing a fault notification scheme. For example, a task might request notification of an event after it has already occurred. PVM immediately generates a notify message in response to any such after-the-fact

request. For example, if a "task exit" notification request is posted for a task that has already exited, a notify message is immediately returned. Similarly, if a "host exit", request is posted for a host that is no longer part of the virtual machine, a notify message is immediately returned. It is possible for a "host add" notification request to occur simultaneously with the addition of a new host into the virtual machine. To alleviate this race condition, the user must poll the virtual machine after the notify request to obtain the complete virtual machine configuration. Subsequently, PVM can then reliably deliver any new "host add" notifies.

The current MPI standard does not include any mechanisms for fault tolerance, although the upcoming MPI-2 standard will include a notify scheme similar to PVM's. The problem with the MPI-1 model in terms of fault tolerance is that the tasks and hosts are considered to be static. An MPI-1 application must be started as a single group of executing tasks. If a task or computing resource should fail, the entire MPI-1 application must fail. This is certainly effective in terms of preventing leftover or hung tasks. However, there is no way for an MPI program to gracefully handle a fault.

IV. Name Service – It is often desirable for two programs to start independently and discover information about each other. A common mechanism is for each of the programs to key on a "well-known name" to look up information in a database. A program that returns information about a requested name is called a name server.

PVM is completely dynamic. Hosts may be added to and deleted from the virtual machine. Processes may start, run to completion and then exit. The dynamic nature of PVM makes name service very useful and convenient. In PVM 3.4, the distributed set of PVM daemons has added functionality to allow them to perform name server functions. In comparison, MPI-1 supplies no functionality that requires a name server. MPI-2 proposes to add functions to allow independent groups of processes to synchronize and create an inter-communicator between them.

Top ten reasons to prefer MPI over PVM –

1. MPI has more than one freely available, quality implementation.

   There are at least <u>LAM</u> and <u>MPICH</u>. The choice of development tools is not coupled to the programming interface.

2. MPI defines a 3rd party profiling mechanism.

   A tool builder can extract profile information from MPI applications by supplying the MPI standard profile interface in a separate library, without ever having access to the source code of the main implementation.

3. MPI has full asynchronous communication.

   Immediate send and receive operations can fully overlap computation.

4. MPI groups are solid, efficient, and deterministic.

   Group membership is static. There are no race conditions caused by processes independently entering and leaving a group. New group formation is collective and group membership information is distributed, not centralized.

5. MPI efficiently manages message buffers.

   Messages are sent and received from user data structures, not from staging buffers within the communication library. Buffering may, in some cases, be totally avoided.

6. MPI synchronization protects the user from 3rd party software.

   All communication within a particular group of processes is marked with an extra synchronization variable, allocated by the system. Independent software products within the same process do not have to worry about allocating message tags.

7. MPI can efficiently program MPP and clusters.

   A virtual topology reflecting the communication pattern of the application can be associated with a group of processes. An MPP implementation of MPI could use that information to match processes to processors in a way that optimizes communication paths.

8. MPI is totally portable.

   Recompile and run on any implementation. With virtual topologies and efficient buffer management, for example, an application moving from a cluster to an MPP could even expect good performance.

9. MPI is formally specified.

   Implementations have to live up to a published document of precise semantics.

10. MPI is a standard.

Its features and behavior were arrived at by consensus in an open forum. It can change only by the same process.

11. Performance depends on the hardware.

However, MPI equipped with more high performance message passing functions than PVM.

12. Number of functions

In the LAM, MPI has 130 functions while PVM has 38 functions.

13. Portability between PVM and MPI

PVM -> MPI is easy. MPI -> PVM is difficult.

Implementation and Definition

One common confusion in comparing MPI with PVM comes from comparing the specification of MPI with the implementation of PVM. Standards specifications tend to specify the minimum level of compliance, while any implementation offers more functionality. In the MPI Forum, many such "added-value" features are listed as expected of a "high-quality implementation". Error handling and recovery are a good example. Standards tend not to mandate specific behavior on errors, other than to list error indicator values. The expectation is that high-quality implementations will give users what they expect.

Specific implementations can easily define their individual handling of errors. Thus, most MPI implementations do not simply abort when an error is detected; just as the PVM implementation does, they attempt to provide a useful error indication and allow the user to continue. Specifically, in any system, there are recoverable and nonrecoverable errors. An example of a recoverable error is an illegal argument to a routine, such as a null-pointer or an out-of-range value. A nonrecoverable error is one where the program may not be able to continue. In many applications, accessing an invalid address or attempting to execute an invalid or privileged instruction is nonrecoverable. The MPI standard does not specify which errors are recoverable, though there has been some discussion in this direction. This is an example of the determination of the MPI Forum to maintain maximum portability—mandating any specific behavior would limit the portability of MPI. Note that even for PVM, some systems provide a less "recoverable" environment than others. For example, systems with proprietary interconnects may kill all processes when any one exits.

Another source of confusion involves features of a particular implementation that are exposed to the programmer. Consider the pvm reg tasker routine that allows a process to indicate to PVM that it, rather than fork/exec, should be used to start tasks. This is an powerful hook to allow extension of the PVM implementation by special applications, such as debugger servers and batch schedulers. MPI, as a standard, has no such object, but specific MPI implementations can and do provide similar services; for example, the MPICH implementation of MPI provides a process startup hook used by the TotalView debugger. The MPI standard does not specify how implementations are to provide this service; as a standard, it should not. At the same time, the experience with TotalView has defined an interface that MPI implementations (not just MPICH) can use, allowing any debugger to access this information. We note that some PVM implementations for massively parallel processors (MPPs) also do not provide the pvm reg tasker routine. This is an example of the freedom of PVM to provide features only in some environments. As a standard, MPI does not have that freedom. If the MPI standard had mandated such a routine, any MPI implementation would have to provide it. Instead, MPI's explicit goals mandated that it choose portability over certain kinds of functionality.

When we compare implementations rather than an implementation of PVM with the MPI standard, the gap in this type of functionality narrows. For example, MPICH, rather than MPI, does provide a way for debuggers like TotalView to access to internal MPICH state on the message queues. Many users want this information, but it raises an interesting issue: How does one define a standard for the internal state of an implementation? For any implementation this can be done, but different implementations may have different internal states. For example, one optimization for communication has the process issuing an MPI RECV send a message to the expected source of the message, allowing the sender to deliver the message directly into the receiver's memory [21]. Should this information be presented to the user? Other implementation choices might eliminate some queues altogether or make it more difficult to find all pending communication operations; in fact, in the MPICH implementation, there is no send queue unless the system has been configured and built to support the message queue service. By not specifying

a model of the internals of an MPI implementation, such as defining a "message queue" does, the MPI standard allows MPI implementations to make tradeoffs between the performance and functionality that the users want.

Future research – PVMPI

The University of Tennessee and Oak Ridge National Laboratory have recently begun investigating the feasibility of merging features of PVM and MPI. The project is called PVMPI and involves creating a programming environment that allows access to the virtual machine features of PVM and the message passing features of MPI.

PVMPI would perform three symbiotic functions –

■ It would use vendor implementations of MPI when available on multiprocessors.

■ It would allow applications to access PVM's virtual machine resource control and fault tolerance.

■ It would transparently use PVM's network communication to transfer data between different vendor's MPI implementations allowing them to interoperate within the larger virtual machine.

Conclusion

The recent publicity surrounding MPI has caused programmers to wonder if they should use the existing de facto standard, PVM, or whether they should shift their codes to the MPI standard. In this paper we compared the features of the two APIs and pointed out situations where one is better suited than the other.

If an application is going to be developed and executed on a single MPP, then MPI has the advantage of expected higher communication performance. The application would be portable to other vendor's MPP so it would not need to be tied to a particular vendor. MPI has a much richer set of communication functions so MPI is favored when an application is structured to exploit special communication modes not available in PVM The most often cited example is the non-blocking send.

Some sacrifices have been made in the MPI specification in order to be able to produce high communication performance. Two of the most notable are the lack of interoperability between any of the MPI implementations, that is, one vendor's MPI cannot send a messages to another vendor's MPI. The second is the lack of ability to write fault tolerant applications in MPI. The MPI specification states that the only thing that is guaranteed after an MPI error is the ability to exit the program.

Because PVM is built around the concept of a virtual machine, PVM has the advantage when the application is going to run over a networked collection of hosts, particularly if the hosts are heterogeneous. PVM contains resource management and process control functions that are important for creating portable applications that run on clusters of workstations and MPP.

The larger the cluster of hosts, the more important PVM's fault tolerant features become. The ability to write long running PVM applications that can continue even when hosts or tasks fail, or loads change dynamically due to outside influence, is quite important to heterogeneous distributed computing.

Programmers should evaluate the functional requirements and running environment of their application and choose the API that has the features they need.

References

1   PVM : Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing. Al Geist, Adam Beguelin, Jack Dongarra.

2   MPI – The Complete Reference. Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, Jack Dongarra.

3   William Gropp, Ewing Lusk. Goals Guiding Design : PVM and MPI.

4   G. A. Geist, J. A. Kohl, P. M. Papadopoulos. PVM and MPI : A Comparison of Features. May 30, 1996.

5   William Gropp, Ewing Lusk. PVM and MPI are Completely Different.

Cost of Software Corporate Users' Perspective

Prof. Ram Verma,

V. N. Bedekar Institute of Management Studies and Research

Abstract

Information technology enjoys the prominent place in today's human activities. In business sector, the importance of IT has increased leaps and bounds. It has tremendously enhanced the velocity of change in every sphere of human life. Computer savvy is a common word now days. In the present business environment one has got to be computer savvy to maintain and improve performance.

When we think of computer, we come across two important peripherals called hardware and software. Whereas hardware is a visible part of the machine, software is generally invisible except that it is in written form. In a sense the software i.e. invisible part is more important than the visible part i.e. hardware since it involves a lot of interface as well as interaction with the user. Further, in terms of costs also, software always supersedes hared ware. The purpose of this article is to look into various cost factors that have significant impact on the decisions relating to the acquisition of a software. The discussion will be confined to the process of ascertaining the cost of sources of soft ware i.e. open source and closed source.

Software

Computer software is a program that enables a computer to perform desired function. It is termed as operating system in general terminology. Again, a soft ware or operating system is further segregated into system software and application software. For example, Windows, Linux are system softwares whereas 'Word', 'Writer' are application softwares.

Sources of softwares

Softwares are generally available from two sources closed sources and open sources.

a) Closed source software

Closed source software (i.e. Microsoft Windows and Office) is developed by a single person or company. Only the final product that is run on your computer is made available, while the all important source code or recipe for making the software is kept a secret. This software is normally copyright or patented and is legally protected as intellectual property. The owner of the software distributes the software directly or via vendors to you the end user. You cannot legally give it away, copy it or modify it in any way unless you have a special license or permission to do so.

Proprietary software is software that has restrictions on using and copying it, usually enforced by a proprietor. The prevention of use, copying, or modification can be achieved by legal or technical means. Technical means include releasing machine-readable binaries only, and withholding the human-readable source code. Legal means can involve software licensing, copyright and patent law. Proprietary software can be sold for money as commercial software or available at zero-price as freeware.

b) Open source software

Open Source software is almost the opposite (i.e. Redhat Linux, Open Office) and is free to use and distribute provided that certain conditions are met.

Free software, as defined by the Free Software Foundation, is software which can be used, copied, studied, modified and redistributed without restriction. Freedom from such restrictions is central to the concept. The usual way for software to be distributed as free software is for the software to be licensed to the recipient with a free software license (or be in the public domain), and the source code of the software to be made available (for a compiled language).

Criteria of selecting a software source

When we come across two types of softwares the question arises which source to select and why. There is no straight way answer to such questions. One has to go through the process of looking into the comparative merits and demerits of both the

options. Broadly, there are two aspects to be considered in this regard:-

    A)Comparative intangible factors of software

    B)Cost of software

A) Comparative intangible factors:

    The following table gives a comparative view of closed source software and open source software:-

Comparative intangible factors of software

CLOSED SOURCE SOFTWARE	OPEN SOURCE SOFTWARE
Totally business oriented approach, profit motive is predominant..	Strongly social oriented approach. Contribution to society is predominant.
Owner prevails over user.	User is free to act.
Source code not available to user.	Source code open to all.
Software is legally protected as copyright or patent.	Software is freely downloadable. Software and authors legally protected.
Support services available.	Support services not available.
Security risk is high.	Security risk is low.
Buyer not allowed to distribute it further.	User can further distribute it.
Software is available against payment only.	Software is available free or with a nominal fee.
Distribution directly by owner or through vendor.	User can download it directly.

    From the comparison it appears that profit motive is predominant in the development and distribution of closed source software and accordingly all parameters are business oriented. In case of open source software, social freedom dimension is in the forefront whereas profit motive is relegated to the back side. All other variation in

functionalities between two type of softwares spring forth from the difference in basic approach i.e. business approach and social approach.

    Once we are clear about various comparative merits and demerits of close and open sources, the next step in decision making process relates to the cost of software.

B. Cost of software

    In this article we are trying to ascertain the cost of software from the perspective of corporate users since because of their large requirements such users have to make considerable investment in softwares which individual users may not require.

    While proceeding to assess the cost of a software, following points need to considered -

   1. What constitutes cost?

    2. Cost to whom?

    Cost of a software includes all the present non recurring investment as well as recurring future expenditure. Cost to whom means the cost of software in the hands of an user. This leads us to the concept of total cost of owning a soft ware or total cost of ownership. The concept of TCO was developed by a research firm Gartner in the late 1980.

The concept of total cost of ownership (TCO)

    As per traditional costing approach, the cost of software may consist of cost of acquisition only but that does not reflect the realistic cost in the sense that a lot of expenditure has to be incurred even after a soft ware has been acquired. Normally, the following activities attract post acquisition costs:-

- installation of software
- making the software operational
- operating costs
- support services costs
- upgradation costs
- retirement costs

    In fact, the post acquisition costs form a large chunk of the total cost. Therefore, while evaluating a software, all these costs have to be logically considered in order to make TCO realistic from the point of view of investment. More over, the product life cycle in IT is generally of shorter duration and this entails further costs in terms of upgradation/
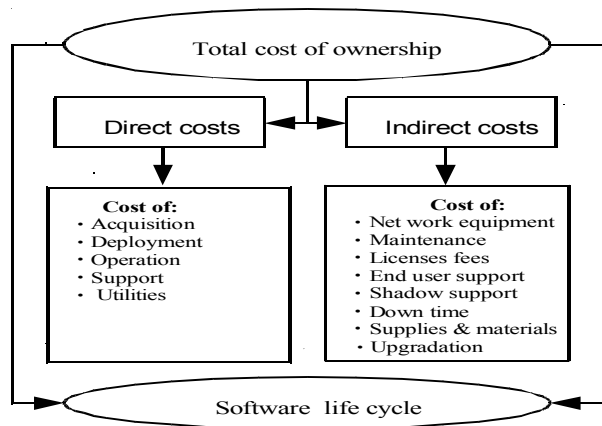
retirements etc. Therefore, product life cycle is a critical aspect and its impact on TCO can not be ignored altogether. The following chart shows various elements of costs in relation to TCO as well as product life cycle:

Further,TCO consists of intangible factors also. Some of the intangible costs are difficult to be quantified in precise terms. For example; cost of time spent on solving computer related problem, cost of security related risks. Yet such intangible factors have a significant bearing on the TCO.

From the above it is observed that a firm must consider the investment in IT from the TCO perspective in order to have realistic picture of the non-recurring as well as recurring expenditure. This will help to implement budgetary process effectively.

Costing methodology

As we have seen above that traditional costing approach does not give realistic view of the total cost of ownership because of short product life cycle, the concept of life cycle costing will be more appropriate in such case. The methodology of cost computation will be as follows:

a) Determine the probable life of the software in terms of years.

b) Identify one time initial costs i.e. cost of acquisition,deployment etc.

c) Identify all future costs i.e. operations, support, utilities, upgradation etc.

d) Discount all future costs in terms of their present value.

e) Total up all initial costs and discounted future costs to get TCO.

Comparative cost of acquisition of software

Based on the above approach, an attempt has been made hereunder to compare the cost of software between closed source and open source. For the closed source Microsoft Windows has been taken as a base, whereas Linux Open Office forms the base for the open source. Various assumptions for computations are mentioned in detail in Annexure - 1. For the sake of simplicity, life cycle and future costs of both the products have been assumed to be the same. Thus the comparison reflects only the initial investment required for the acquisition of the software.

The analysis is based on a typical factory having an annual turnover of Rs.200 crore with an employee strength of 150 at a single location. The computations are based on the requirement of 100 desktops.

Comparative cost of acquisition

Sr. no.	Particulars	Closed source software	Open source software
1	Number of desktops	100	100
2	Operating system	Microsoft Windows	Linux Open Office
3	Applications	Windows based	Linux based
4	Cost of acquisition*	42.37 lakh	0.50 lakh

● please refer Annexure 1 and Annexure 2.

Decision making

As stated in para 4 above, the decision process calls for the consideration of two factors .i.e. intangible factors and tangible factors i.e. cost factors. The relative merits of open source as mentioned in para 4(a) clearly outweigh its demerits as compared to closed source considered along with the cost factor that heavily leans in favor of the Open Source.

This may be only a tip of iceberg. Real life situations will definitely call for the computation of the entire cost of ownership or TCO to support the decision making process. Alongside, the factors such as the mindset of employees, resistance to change and the resulting cost aspects have also to be looked into to make effective decision for investment in IT.

Conclusion

Management of costs is vital for the growth and survival of any organization. IT is one of the important areas where an organization has considerable scope for managing costs in terms of both the present and future environment. With the emergence of the concept of the total cost of ownership, the total outgo in IT can be estimated realistically to a considerable extent. In today's widely distributed IT computing environment, it becomes more pertinent to understand TCO in order to effectively evaluate all the deployment alternatives. Along with this, the comparative analysis of investment requirements from both the options i.e. closed source as well as open source is also quite essential to save on the opportunity costs and thus rationalize the utilization of the financial resources.

Details of cost information & assumptions

1.Cost of operating system:

A) Common to all depts

Operating system	Approx. cost (Rs.)
STD MS Office	10500/- per seat
XP Professional	7200/- per seat
Prof. MS Office	16000/- per seat

B) Cost of data base: Rs.11000/- per user (Oracle)

C) R & D dept. (additional sotware)

Auto CAD : Rs. 121000/- each seat

Photo shop : Rs.36000/-

Photo shop suite (one) : Rs. 50000/-

2.Cost of server

a) Fixed cost : Rs. 36000/- (with 5 user licenses)

b) For each additional sheet: Rs. 2100/- extra

c) Total cost for 20 seats : Rs. 68000/- (Rs. 36500+Rs.2100*15)

3.Requirements of closed source operating system in a typical factory

Assumptions

A. General information
a) Annual turn over: Rs. 200 crores

b) Single location
c) No. of desk tops: 100
d) No. of employees: 150

B. specific information

1.Distribution of department-wise desktops:

- Accounts dept.   :   10
- Purchase and stores   :   15
- Shop floor   :   20
- Adm. and HRD   :   10
- Sales   :   20
- R & D   :   10
- Others   :   15

  Total desktops   100

2. Distribution of servers

- Shop floor: one
- Adm. and HRD: one
- R& D : one
- Accounts/sales / Purchases:one
- Total servers : four

3. Data base requirement

Distributed as per number of seats department wise.

References

1. http://www-1.ibm.com/linux/RFG-LinuxTCO-vFINAL-Jul2002.pdf

2. http://www.phptr.com/articles/article.asp?p=24404&rl=1

3. http://www.opensource.org/docs/definition.php

4. http://opensource.mit.edu/papers/cominomanenti.pdf#search=%22why%20closed %20source%22

5. Shri Rajagopal Iyer

6. Shri Kastubh Kale

Summary of cost of proprietary software (Closed source)

Software\Depts.		Accounts	Adm.& HR	Org.wide	Others	Purchase	R&D	Sales	Shop Floor	Total Result
Antivirus	Sum - Seats			106						106
	Sum - Amt. (Rs.)			127200						127200
AutoCAD	Sum - Seats						5			5
	Sum - Amt. (Rs.)						605000			605000
CAL	Sum - Seats	10	10		15	15	10	15	15	90
	Sum - Amt. (Rs.)	21000	21000		31500	31500	21000	31500	31500	189000
D.B. Client	Sum - Seats	10	10		15	15	10	20	20	100
	Sum - Amt. (Rs.)	110000	110000		165000	165000	110000	220000	220000	1100000
MS Exchange Server	Sum - Seats			1						1
	Sum - Amt. (Rs.)			28000						28000
Off. Professional	Sum - Seats	3	3		5	5	3	5	3	27
	Sum - Amt. (Rs.)	48000	48000		80000	80000	48000	80000	48000	432000
Off. STD.	Sum - Seats	7	7		10	10	7	15	7	63
	Sum - Amt. (Rs.)	73500	73500		105000	105000	73500	157500	73500	661500
Photoshop	Sum - Seats						5			5
	Sum - Amt. (Rs.)						180000			180000
Photoshop Suite	Sum - Seats						1			1
	Sum - Amt. (Rs.)						50000			50000
Sever	Sum - Seats		1				1	1	1	4
	Sum - Amt. (Rs.)		36000				36000	36000	36000	144000
XP Professional	Sum - Seats	10	10		15	15	10	20	20	100
	Sum - Amt. (Rs.)	72000	72000		108000	108000	72000	144000	144000	720000
Total Seats		40	41	107	60	60	52	76	66	502
Total Amt. (Rs.)		324500	360500	155200	489500	489500	1195500	669000	553000	4236700

OS for Embedded Systems: An Introduction to Embedded Linux

Linux Thane 2006

24th Nov, 2006

J A Gokhale

WinVision

Why OS?

- To run one single program is easy
- 2, 3 or 4 may be fine. But more than 5 is difficult without help
- Need to take care of
  - Memory
  - Program counters
  - Scheduling (run how many instructions from each?)
  - Communications
  - Synchronization
- OS enables us manage all these.

Agenda

- Introduction to OS concepts
- Real Time Systems : Concepts
- Embedded Systems Development Cycle
- Embedded OS: Case Study
  - RTLinux
  - VxWorks
- Q & A

What is an Operating System?

- Provides environment for executing programs
- Manages and schedules all resources : Process abstraction for multi tasking and concurrency
- Hardware abstraction : device drivers
- File systems support
- Communications & Networking
- User Mgmt
- We will focus on concurrency and real time issues

Real Time Systems: Concepts

OS Flavors

- Desktop
  - Windows (9X, XP Home, XP/2000 Pro)
  - Mac
- Server
  - Windows (XP/2000 Server &Advanced Server)
  - Unix Varieties
- Embedded
  - Many

What is an Embedded OS?

- An embedded operating system runs on any embedded platform, and specifically built for the cause.
- The OS is loaded as software in ROM of a single-board microcomputer .
- Embedded applications start running special purpose programs at power on and will not stop until turned off.
- These applications will not usually have any peripheral support (keyboard, monitor, serial connections, mass storage, etc.) or a user interface.
- Often an embedded OS must provide real-time response to perform its requirements.

Do I need one?

- Not always
- Simplest approach : use a cyclic executive

```
for ( ; ; ) {
    do part of task 1;
    do part of task 2;
    do part of task 3;
}
```

Kernel Types

- Kernel: Smallest portion of the OS which provides task scheduling, dispatching, inter task communication
- Kernel types
  - **Nanokernel – the dispatcher**
  - **Microkernel – nanokernel with task scheduling**
  - **Kenel – microkernel with intertask synchronisation**
  - **Executive – a kernel with privatised memory blocks, I/O services, etc. Most RTOS fall in this category**
  - **OS- the executive that also provides user management, file system management, security etc**

Nano-kernel: Single task cyclic executive

```
setup-timer();
c=0;
while (1) { suspend until timer expires
  c++;
  compute tasks due every cycle;
  if ((c%2)==0) compute tasks due every 2nd cycle;
  if ((c%3)==0) compute tasks due every 3rd cycle;
}
```

What makes a good Embedded OS?

- Modular
- Scalable
- Configurable
- Small footprint
- CPU support
- Device drivers
- etc, etc, etc...

Cyclic Executive

- Advantages
  - Simple implementation
  - Low overhead
  - Very predictable
- Disadvantages
  - Cannot handle sporadic events
  - Things should happen in lock step
  - Code must be scheduled manually

Interrupts

- Environmental events that demand attention
  - A byte arrived on a communication channel
  - Sensor recorded the current temperature
- ISR is executed in response to the interrupt

Can the executive support interrupts???

- Solution: Cyclic Executive + Interrupts
  - Works fine for many low end applications
  - Cheap and predictable interrupt handler
  - No context switch, no environment save

Main functions of an OS

- Process management
- Memory Management
- Interrupt Handling
- Exception handling
- Disk Management
- Process synchronization (IPC)
- Process scheduling

Drawbacks of CE + Interrupts

- Programmer is responsible for scheduling
- Programmer is responsible for interrupt prioritizing
- Scheduling is static
- Sporadic events handled slowly

Processes, Threads & Tasks

- A process is a program in execution
  - Starting a process is a heavy job for the OS as it has to allocate memory, swap code, manage data structures
- A thread is a light weight process
  - Several threads share the same memory, address and data structures, global and static variables, heap
  - They have their own PC, program status, registers, stack
  - Shorter creation and context switch time than processes
- A task is mostly a thread

Cooperative Multi tasking

- Cheap alternative
- Non preempting
  - Process / task runs to its full length
- Processes responsible for relinquishing control
  - Errant program would lock up the entire system

Real Time Systems
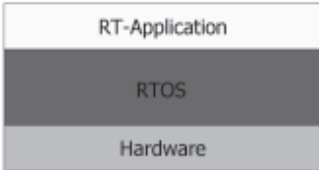
What is Real Time?

"A real time system is one in which the correctness of the computations not only depends upon the logical correctness of the computation but also upon the time at which the result is produced. If the timing constraints of the system are not met, system failure is said to have occurred."

- Donald Gillies

Variants of RTOS

- Pure real time OS

| RT-Application |
| RTOS |
| Hardware |

Definition

- What is a Real-Time Operating System?
  - Real Time operating system is an operating system that guarantees specific response times to events. (Internal and External)
    - The response usually falls within some small upper limit of response time (typically milli- or micro-seconds).

Pure Real Time OS

- Especially designed for real-time requirements
- Completely real-time compliant
- Often usable for simple architecture
- Advantage: no or little overhead
  - Computing power, memory
- Disadvantage: limited functionality
- Example: eCos, Nucleus, pSOS, VxWork, QNX, OSE, Lyra

Definition

- Types of Real-Time Operating System?
  - Soft Real-Time: Processing events late or missing one **DOES NOT** causes catastrophic consequences.

  - Hard Real-Time: Processing events late or missing one **DOES** causes catastrophic consequences.

OS Real-Time Extensions

- Extension of an OS by real-time components
- Cooperation between RT- and non-RT parts
- Advantages: rich functionality
- Disadvantage:
  - No general real-time ability
  - Computing and memory resources
- Example: RT-Linux, Solaris, Windows NT

OS Real-Time Extensions
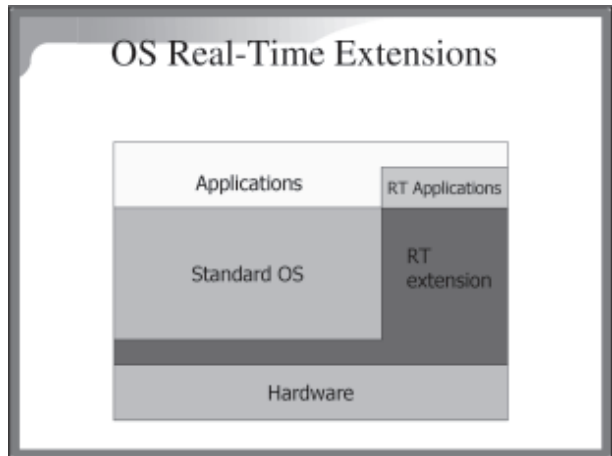
Task: basic notation in RTOS

- Task = thread. (light weight process)
  - A sequential program in execution
  - It may communicate with other tasks
  - It may use system resources
- May have timing constraints

RTOS-Kernels

Real-time Kernel

| application software |
| middleware | middleware |
| device driver | device driver |
| real-time kernel |

General Purpose OS

| application software |
| middleware | middleware |
| operating system |
| device driver | device driver |
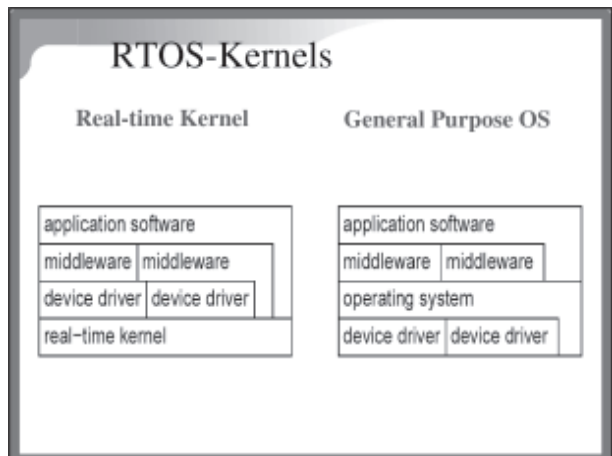
Task classification

- Hard RT - must be completed within the given deadline
- Soft RT - task may be completed after the deadline
- On time RT tasks : alarm, triggers etc
- Non RT tasks (background tasks)

Basic functions of RTOS kernel

- Task Management
- Interrupt handling
- Memory management (no VM for hard RT)
- Exception handling
- Task synchronization
- Task scheduling
- Time management

Task states

- Ready
- Running
- Blocked (waiting)
- Idling
- Terminated

Real-time Operating System

- Functions:
  - task management,
    - scheduling, dispatcher
    - communication (pipe, queue)
    - synchronization (semaphore, event)

Exception handling

- Exceptions, (ex: missed deadlines, out of memory, time out, etc)
- Exception handler
  - **System call with error code is reported**
  - **System executes a special code to handle the error**
    - Take default values
    - Reset event registers, counters, flags etc
  - **Watchdog**
    - A thread which runs at the highest priority parallel to the other threads which reacts when exceptions are recognised

Interrupt Handling

- Asynchronous and Synchronous (software generated)
- Interrupt enable/ disable
- Priority
- Nesting
- Chaining
- Sharing across several devices (OR tied)

Task Synchronization

- Use of
  - **Semaphores**
    - Counting, or binary. Task gets it only when semaphore is 0
  - **Mutex**
    - Binary semaphore, associated with a owner
  - **Spinlock**
    - Lock mechanism for multiprocessor systems
  - **Read/Write locks**
    - Protection from concurrent write, allowing concurrent reads
- Critical section, race condition, deadlock!!

Memory Management

- Block , paging
- Many embedded OS do not have memory protection
- Tasks may access any block!!
  - Caution….
- Some commercial RTOS provide memory protection
  - Run to 'safe mode' when illegal access trap occurs

Task Scheduling

- Scheduler is responsible for time sharing of CPU among tasks
- A schedule is an ordered list of tasks to be executed, and the schedule is feasible if it meets all the deadlines

Real time

- Main goal of a RTOS scheduler is to meet deadlines
  - If you have 4 homeworks, and one is due in an hour, pick this one!!!
  - Based on the criticality / priority assigned

Priority based Preemptive scheduling

- Always run the highest priority runnable process

Priority based scheduling

- Typical RTOS has a fixed-priority preemptive scheduler.
- Assign each process a priority (done once, and no changed!)
- At any time, scheduler executes the highest priority process ready to run (processes can be blocked, waiting for resources)
- Process runs to completion unless preempted

Rate monotonic scheduling

- Common way to assign priorities
- Processes with shorter period are given higher priority

Period	Priority
10	1 (high)
12	2
15	3
20	4 (low)

RTOS task model

- Each task is a triplet: (exec time, period, deadline)
- Can be initiated anytime during the period

$p = (2,5,5)$

EDF Schedule

- RMS assumes fixed priorities
- Can we have dynamic priorities?
- Earliest Deadline First
  - Processes with soonest deadline given highest priority

$p_1 = (2,4,4), p_2 = (3,6,6),$ 100% utilization

p_2 takes priority with its earlier deadline

EDF drawbacks

- EDF is complicated
- Overhead for dynamic management
- Cannot guarantee which process will run when

Embedded Systems Development Cycle

Priority based scheduling in RTOS

- Static priority
  - Task given priority at the time it is crated
  - Scheduler picks up the task with highest priority from the wait Q
- Dynamic priority
  - Task priority is calculated online
  - EDF – task with a closer deadline gets higher priority
  - RMS – a task gets more priority if it has to run more frequently

Embedded System Development Cycle

- Development process
  - integrate HW components
  - develop programs
  - test

Time Management

- A high resolution hardware timer is programmed to interrupt the processor at fixed rate
- Each time interrupt is called a tick; used as the basic unit of time measure
- Use this interrupt to update timer counters, clocks etc

Embedded System Dev Env

- Development environment and development pipeline

Development Environment

- Use the host to
  - edit, compile, and build application programs
  - configure the target
- At the target embedded system, use tools to
  - load, execute, debug, and monitor (performance and timing)

Three Concurrent Tasks Within a Programmable Thermostat

/* Monitor Temperature */	/* Monitor Time of Day */	/* Monitor Keypad */
do forever {	do forever {	do forever {
measure temp ;	measure time ;	check keypad ;
if (temp < setting)	if (6:00am)	if (raise temp)
start furnace ;	setting = 72°F ;	setting++ ;
else if (temp >	else if (11:00pm)	else if (lower temp)
setting + delta)	setting = 60°F ;	setting-- ;
stop furnace ;	}	}
}		

Simple SW Structure for Embedded Systems

- To write the control software (program) for a smart washer
  - initialize
  - read keypad or control knob
  - read sensors
  - take an action
- System current state
  - state transition diagram
  - external triggers via polling or ISR
- Threads for concurrent operations

Concurrency

- Only one thread runs at a time while others are *suspended*.
- Processor switches from one thread to another so quickly that it appears all threads are running simultaneously. Threads run *concurrently*.
- Programmer assigns *priority* to each thread and the *scheduler* uses this to determine which thread to run next

Program Organization of a Foreground/Background System

Real-Time Kernel

- Threads call a library of run-time routines (known as the real-time *kernel*) manages resources.

- Kernel provides mechanisms to switch between threads, for coordination, synchronization, communications, and priority.

Context Switching

- Each thread has its own stack and a special region of memory referred to as its *context*.
- A *context switch* from thread "A" to thread "B" first saves all CPU registers in context A, and then reloads all CPU registers from context B.
- Since CPU registers includes SS:ESP and CS:EIP, reloading context B reactivates thread B's stack and returns to where it left off when it was last suspended.

Programming Languages Used in New Embedded Designs

The build and load process for desktop application programs.

Processor Types Used in New Embedded Designs

The build and load process for embedded application programs.
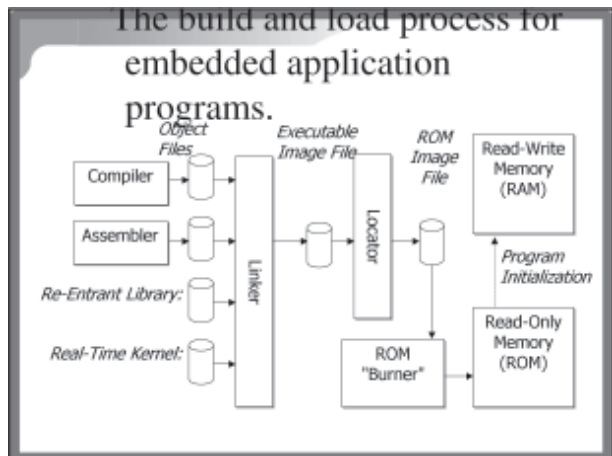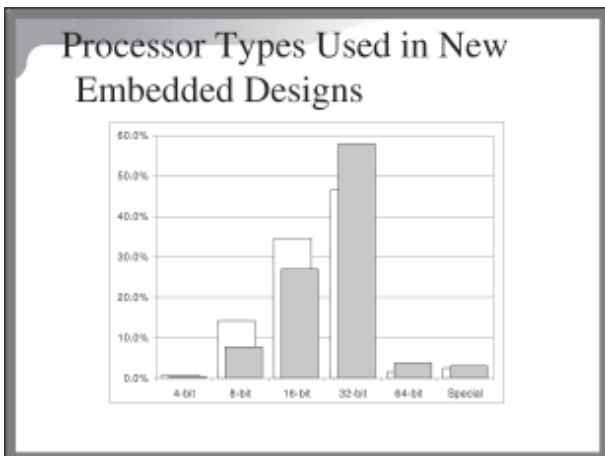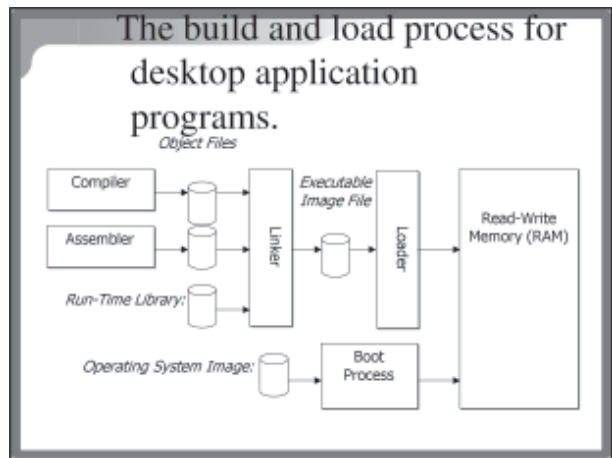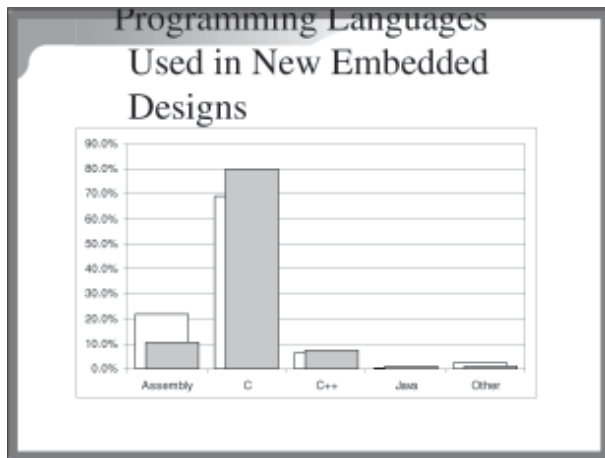
Embedded OS: Case Study

What makes a good RTOS?

- Multi-threaded and pre-emptible
- Thread priority has to exist because no deadline driven OS exists
- Must support predictable thread synchronization mechanisms
- A system of priority inheritance must exist

Microsoft

- Embedded NT/XP
  - "Real-time" control
- Windows CE (CE.NET)
  - Internet devices
- Pocket PC 2003
  - Handheld PC's and PDA's

Who are the Embedded OS players?

- Wind River Systems
  - VxWorks
  - pSOS
- QNX Software Systems
  - QNX
- Green Hills Software
  - Integrity

Commercial Embedded Linux

- FSMLabs - Open RT Linux
- AMIRIX Embedded Linux
  - derived from Debian
- Coollogic Coollinux
  - combines Linux and Java for Internet apps
- Lineo Embedix
  - supports real time and high availability apps
- MontaVista Linux
  - general purpose embedded solution

Who are the Embedded OS players?

- Palm Computing
  - PalmOS
- Symbian
  - SymbianOS

Commercial Embedded Linux

- Red Hat Embedded Linux
  - general purpose embedded solution
- TimeSys Linux GPL
  - low latency enhanced kernel
- Vital Systems vLinux
  - for ARM based embedded apps

Open Source Embedded Linux

- Embedded Debian Project
  - convert Debian to an embedded OS
- KURT - event schedules with 10us resolution
- uCLinux
  - for microprocessors that don't have MM
- uLinux (muLinux)
  - distro fits on a single floppy

A Typical Embedded Linux

µ Browser	Pocket Word	ICQ	E-mail	MP3	PIM	UserApp
JavaVM		GTK+ & GDK GUI				
SDK	DDK	GW32			Multi-Language	
Embedded Linux OS						

PDA

Why Use Linux?

- Open Source
- Reliability
- Scalability
- Portability
- Supports Virtually All Network Communication Protocols
- Initial startup costs are very small
- Royalty free

RT Linux

- RTLinux is an OS, in which a small real time kernel co-exists with standard Linux kernel
  - RT kernel sits between the Linux kernel and the hardware
  - RT kernel intercepts all interrupts
    - Real time interrupts are processed by the RT kernel, others are passed to the Linux kernel as s/w interrupts
    - Only for the RT interrupts trapped by the RT kernel, appropriate ISRs are run

What is Embedded Linux?

- The Linux OS ported to an embedded system.
  - Generally contain a smaller subset of functionality.
    - Less services provided.
    - Less memory required.
    - Boots from ROM.
    - No keyboard or mouse required.
    - Special software developed to control embedded peripherals. (flash disks, touch screens, tiny displays)

RT Linux

- RT kernel assigns the lowest priority for the Linux Kernel, so that all events will be executed in real time
- User can create real time tasks and can decide on the scheduling algorithms, priorities, execution frequencies etc
- Real time tasks are privileged, they have direct access to hardware registers. They o not use VM.

RT Linux Model
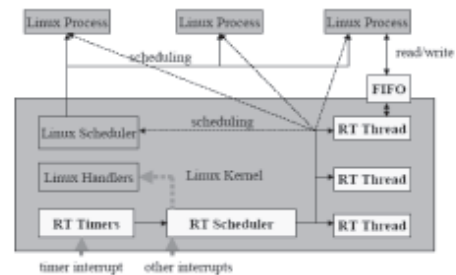
Embedded Linux Tools

- Java
  - VisualAge from IBM
- Embedded Graphics Tools
  - Embedded Qt*
- Embedded Database Tools
  - Sleepycat*

Scheduling in RT Linux

- RT Linux allow different schedulers
  - EDF
  - Rate monotonic scheduler
  - Fixed priority scheduler

uClinux

- uClinux is not Linux, but an embedded version
- Runs on CPU's which do not have memory management. Uses a flat memory model.
- No difference between user space and kernel space
- All applications run at privilege level 0
- All tasks have direct access to memory, I/O devices, and timers.

Embedded Linux Tools

- Standard GNU tool set is available from OS providers.
- Micro Cross*
  - **GNU X-Tools™ includes:**
    - C/C++ Compilers, Assemblers, Linkers, more…
    - V Integrated Development Environment
    - GDB Debugger and stubs for Ext. Debugging
    - Simulators for most targets
    - All Source Code Included -- GPL License

VxWorks

- VxWorks is a commercial hard real time operating system developed by wind river systems.
- The main idea: use monolithic kernel to schedule user tasks according to user defined priorities. Maximize kernel timing predictability.
- Gives the users maximal control.

VxWorks

- A dedicated real time system, not intended as a general purpose OS.
- lacks many modern OS features that interfere with real time performance (flat memory model, no paging).

VxWorks - limitations

- Lacks many modern OS features.
- Guaranteeing the deadlines is the responsibility of the user at design time.
- Doesn't support most modern applications and APIs (only a small subset of POSIX).
- Despite the flat memory model dynamic memory allocation still cases memory fragmenting, which increases timing unpredictability.

VxWorks

- Scheduling is done using a preemptive priority driven approach, priorities are chosen arbitrarily by the user (0-256).
- Priorities can be changed by the user at runtime but this is discouraged.
- A user can lock a task so that it can't be preempted even by higher priority tasks or interrupts.
- This allows the use of the fixed priority response time analysis to check schedulability offline.

Q & A
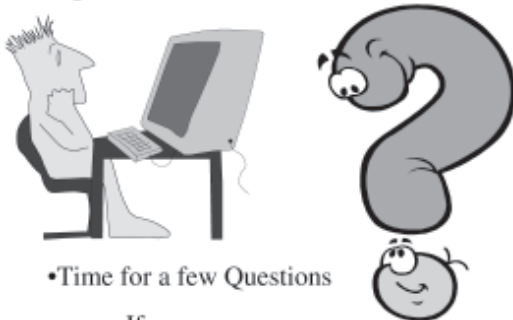
- Time for a few Questions
- If none, ...

VxWorks

- Is resource sharing aware and has a priority inheritance built in.
- Optimizations in implementation of the context switch and the return from interrupts.
- The kernel never disables NMI (non-maskable interrupts) so they are always available to the user.

THANKS

The greatest difficulty in the world is not for people to accept new ideas but to make them forget about their old ideas.

- John Maynard Keynes

Open Source For Library: A Case Study

Priyanka S. Kadam, Yashada V. Naik

S.Y.B.Sc. (I.T.), B. N. Bandodkar College, Science, Thane

Abstract

Information is a term with many meanings depending on the context, knowledge, instruction, communication, representation and mental stimulus. Information should be sufficient, competent, relevant and useful. Therefore it has been described as the fifth need of man ranging after air water, food, and shelter. The main sources of information are books and therefore we maintain Library. A digital library is the greatest revolution, which extends and enhances the existing information storage and retrieval. Present paper talks about how Open Source softwares will be useful for creation of digital libraries. If you've ever used the Internet, you've used open source software.

Library, in Brief

The main sources of information are books and therefore we maintain Library. Library is the place where we find the collection of various types of books for the purpose of Reading, Studying and for the reference. According to Dr. S. R. Rangnathan, who is the father of Library and information science in India, Library is the public Institution or establishment charged with the care of a collection of books and the duty of making them accessible to those who require use of them. As far as working of Traditional Library concern user must go to the Library and search the information, which is time consuming. Therefore it is hardly necessary to have a facility to access information where user sited. Thus need of digital library emerged. A digital library is the greatest revolution, which extends and enhances the existing information storage and retrieval. The purpose of Digital Library is to provide fast, un-interrupted access to the resources through INTRANET as well as INTERNET. Digital Library can be accessed through browsers. Traditional libraries are limited by storage space but digital libraries have the potential to store much more information, simply because digital information requires very little physical space to contain it. the cost of maintaining a digital library is much lower than that of a traditional library. A traditional library must spend large sums of money paying for staff, book maintenance, rent, and additional books. Digital libraries do away with these fees. Multiple users can access it at a time, and therefore availablity of books increases. Users can access it round a clock. Using networking approach we can connect number of libraries and using this fascility users can refer different types of books which is not possible in traditional Libraries, because in traditional Labraries users or readers can refer only those books which are present in that particular Library.

Open Source

Open Source Software is a computer software whose source code is available under a copyright license that permit users to study, change, and improve the software, and to redistribute it in modified or unmodified form.

Open Source Library software enpower users, particularly in universities, Libraries and other public service institutions, to build their own digital libraries. Open source software gives orgnisation flexible tools for managing and delivering their digital content. Open Sourse Software is a suite of software for building and distribuiting digital Library collections. It provides a new way of orgnizing information and publishing it on the internet or on CD-ROM. It encourges the effective deployment of Digital Libraries to share information and place it in the public domain.

If you've ever used the Internet, you've used open source software. Many of the servers and applications running on machines throughout the wired world rely on software created using the open source process. Example of such software is Apache.

Library Software Today

No software is perfect. There is constant innovation in library software. For many of us online catalog systems mean a clunky old text interface that often is less effective than browsing stacks. Often, this is due to the obstacles we face in managing legacy systems; new systems might be vastly improved, but we are slow to upgrade when we consider the costs of migrating data, staff retraining, systems support,

and on and on. Sometimes, new versions of systems we currently use are just not good enough to warrant making a switch.

This is not surprising. The library community is largely made up of not-for-profit, publicly funded agencies, which hardly command a major voice in today's high tech information industry. Online systems are no less about access to information than having an auto-open front door or an elevator in a library building.

We read of exciting technological innovations in library-related systems. Innovations in advanced user interfaces and metadata-enabled retrieval environments and other areas have the potential to make online access more and more seamless and easy to use.

Libraries might do well to enhance their services by leveraging community-owned information systems-which open source seems to promise.

First, open source systems, when licensed in the typical "general license" manner, cost nothing (or next to nothing) to use-whether they have one or one thousand users. Although the costs of implementing and supporting the systems on which software runs might not change, imagine removing the purchase price of a new search interface (or ILL tool, or circulation module, etc.) from your budget for next year. Rather than spending thousands on systems, such funds might be reallocated for training, hiring, or support needs, areas where libraries tend toward chronic shortfalls.

Second, open source product support is not locked in to a single vendor. The community of developers for a particular open source product tends to be a powerful support structure for Linux and other products because of the pride in ownership described above. Also, anyone can go into business to provide support for software for which the very source code is freely available. Thus even if a library buys an open source system from one vendor, it might choose down the road to buy technical support from another company-or to arrange for technical support from a third-party at the time of purchase. On top of this flexibility, any library with technical staff capable of understanding source0020code might find that its own staff might provide better internal support because the staff could have a better understanding of how the system work.

Third, the entire library community might share the responsibility of solving information systems accessibility issues. Few systems vendors make a profit by focusing their products on serving the needs of users who cannot operate in the windows/icons/menus/pointer world. If developers building systems for the vision impaired and other user groups requiring alternative access environments were to cooperate on creating a shared base of user interfaces, these shared solutions might be freely built into systems around the world far more rapidly and successfully than ever before.Beyond merely using open source products, however, we must create them. For those of you who realize that someone else might benefit from what you've done-and that you might benefit from the ability to share in the work of others-consider thoroughly the implications of releasing your code under an open source license.

Annexure I

Some Open Source Softwares For Library:

Library software provides libraries throughout the world with a tool to management the vast amount of information contained in a broad spectrum of mediums-from traditional books to microfiche to electronic media. We have gathered leading library software solution providers for you to review.

Koha

Koha is open source software. The librarian interface is tested only with Mozilla/Firefox.Several companies around the world support Koha, providing libraries with a full array of vendor services including installation, migration assistance, data integrity testing, staff training, software maintenance, support and customization. http://prdownloads.sourceforge.net/koha/koha-2.2.2.tar.gz?download

Greenstone

is a suite of software for building and distributing digital library collections. It is not a digital library but a tool for building digital libraries. It provides a new way of organizing information and publishing it on the Internet in the form of a fully searchable, metadata-driven digital library. It is open-source, multilingual software, issued under the terms of the GNU General Public License. The survey received 62 valid responses from users and developers who work with Greenstone in at least 32 different countries. www.greenstone.org/cgi-bin/library

Dspace

The DSpace digital repository system captures, stores, indexes, preserves, and distributes digital research material. Research institutions worldwide use DSpace as an institutional repository, a learning object repository, for records management, and more. The DSpace open source platform is freely available so you can customize and extend it to suit your needs. DSpace runs on any UNIX or LINUX operating system. http://www.dspace.org/

Fedora

Fedora open source software gives organizations a flexible service-oriented architecture for managing and delivering their digital content. At its core is a powerful digital object model that supports multiple views of each digital object and the relationships among digital objects. Digital objects can encapsulate locally-managed content or make reference to remote content. Dynamic views are possible by associating web services with objects. Digital objects exist within a repository architecture that supports a variety of management functions. All functions of Fedora, both at the object and repository level, are exposed as web services. These functions can be protected with fine-grained access control policies.This unique combination of features makes Fedora an attractive solution in a variety of domains. Some examples of applications that are built upon Fedora include library collections management, multimedia authoring systems, archival repositories, institutional repositories, and digital libraries for education.

Libman

Library Manager, as you could understand from its name, is a library management program. You can easily take control all of the books under any circumstance. This could be in a school library or in a home library that we all have.

References

1) Annals of Library and Information Studies National Institute of Science Communication And Information Resources, CSIR

2) www.med.yale.edu/library/oss4lib

    www.med.yale.edu/library/oss4lib

    oss4lib@biomed.med.yale.edu.

Experimentation of LINUX on VPM Campus

Abhijeet A. Kale[1], Dalvir Reel[2],

1. Prof. In-charge, Department Of Information Technology, B. N. Bandodkar College Of Science, Thane

2. System Analyst, Department of Information Technology, VPM's Polytechnic, Thane

Abstract

The goals of education and technology align so closely that we can have profound impact on the way we think and learn. Technology can help to make the education easier, simpler. By observing the benefits of computers in education, Vidya Prasarak Mandal (VPM), a premier educational trust in Thane, Maharashtra started introducing computer education in all illustrious, model institutions run by them. Initially VPM selected the proprietary softwares because of the helping nature. In September 2004, VPM started thinking of alternatives came across Linux / Open Source system. Present paper says why VPM is required to change their platform and how we are switching to Linux.

Computers on VPM Campus
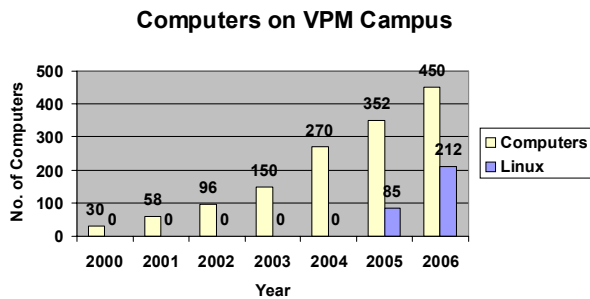
From 2002, VPM started gaining importance of computers and made computers available for students. Initially all computers were Windows based. In 2004, we were spending significant amount for license per year although it is used for education purpose. More ever Government / University does not give any grant for this amount. It's a heavy burden on management. VPM started thinking of alternative. We come across Linux / Open Source and started working on this platform. We started converting our systems to Linux / Open Source platform. We are trying to do day-to-day activities on Linux, although the progress is slow. Currently Linux is proving more than up to the task. And in those few places where Linux appeared to under whelm, we are finding different ways to accomplish the task.

The institutions run by VPM started playing major role in making the project success.

B. N. Bandodkar College of Science, one of the institutions belonging to VPM, played big role by making one of the IT laboratory consisting of 25 computers and at least one computer in every department purely Linux based. Out of 110 computers, currently 37 computers are Linux based

and teaching as well as non-teaching staff is working or learning on it. IT department started conducting practicals of F.Y.B.Sc., S.Y.B.Sc. on Linux and students started finding Linux also user friendly like Windows.
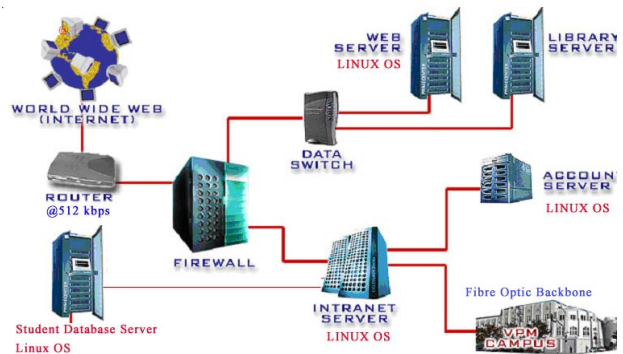
Current percentage of Linux based Computers on VPM Campus
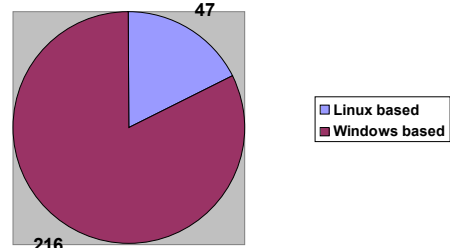
VPM's Polytechnic, another institution of VPM, converted their one IT laboratory on Linux OS while two other laboratories dual boot. They are conducting practicals of T.Y.I.F, T.Y.CO. on Linux platform.

Vidya Prasarak Mandal, on it's part, has deployed Linux based Thin client network with around 100 thin clinets catering to over 200 stakeholders in our management institute namely V.

N. Bedekar Institute of Research and Management, for day to day use. Though there were some teething problems, the end users started using it and are in fact quite happy with the performance, stability and privacy it provides.

Why not 100 % conversion ratio?

VPM's institutes are affiliated to University of Mumbai, M.S.B.T.E., Y.C.M.O.U. The syllabus for all classes is framed in such way that there is little chance to work on Linux platform. Windows is necessity for maximum of the syllabus.

Here's the approximate process we are using to transit from a Windows to Linux desktop. Of course, our mileage may vary:

➢ Ban proprietary upgrades and new proprietary programs immediately

➢ Segregate your data from your apps and OS

➢ Make a transition plan

➢ Find dead ends in your plan, and find and ways out of those dead ends

➢ Create a practice setup

➢ Get hardware ready to accept Linux

➢ Install Linux

➢ Do final windows backup

➢ Test to verify a working setup

➢ Do a backup of your Linux data

➢ Back up

Out of 17 different servers on campus for different purposes, 7 are Linux based while 6 are dual.

Current percentage of Linux based Servers
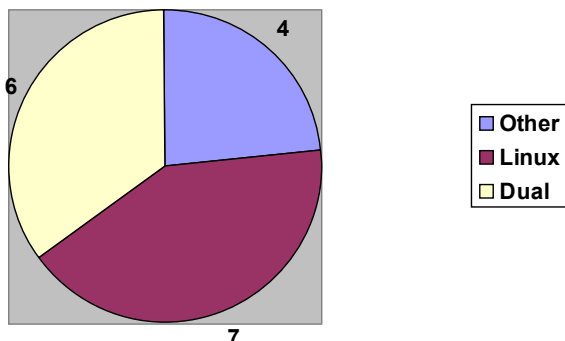
Along with Conversion from Windows to Linux, VPM has also undertaken another two projects viz**.**

Linux based Digital Library

Marathi E-books.

Problems we are facing currently:

Drivers, Drivers, Drivers: Linux either is not able to detect some piece of hardware or is not able to assign a workable driver to it. Very difficult to find a Linux driver for the device at hand.

Windows has a standard process for installing and uninstalling programs. This is a Linux weakness.

Windows offers consistent, easy to understand, and reliable GUI (graphical user interface) controls for managing hardware configuration and other settings—such as boot options, audio, video, and screen resolution.

Support For The Latest, Greatest Technologies: Even though Windows users have tended to grumble about the lack of built-in support for newer technologies, such as DVD-burning, Microsoft has done an admirable job of supporting new technologies over the years. For example, WinXP supports USB (Universal Serial Bus) 1.x, USB 2.0, FireWire, CD-RW, and a long list of others. Linux has not kept pace with this, although some distros are showing signs of solid improvement.

The major difference between Linux and Windows is the open-source nature of Linux and closed-source nature of Windows. The user as he/she sees fit can modify Linux, as an open-source OS. Open source really means that anyone can modify the underlying code, and that no one completely owns it. Microsoft Windows, on the other hand, is not an operating system you can modify easily - it was designed to work a particular way with a certain user interface as designated by Microsoft software engineers.

Conclusion:

1. Government / University must be asked to modify the syllabus in order to make educational institutions free from License copy overheads.

2. They at least should not insist to work on specific platform. Freedom should be given to institutions for there own choice.

LINUX POWERED

ROOT ®

Linux-Consulting Training

Root Technologies

nurture freedom, celebrate independence

ROOT ®

Linux-Consulting Training

POWERED BY RED HAT

Notes