FOSS Nepal
सफ्टवेयर स्वतन्त्रता

THE init Magazine

Issue #8
June '10

EDITORIAL

We feel very proud to bring to you yet another issue of 'The init Magazine'. We are really overwhelmed by the response to the revival of the magazine. We had wonderful feedback for our last issue and we hope to cater good content in our coming issues as well.

This issue is packed with several interesting articles. A new version of Fedora was released a few weeks ago. We bring you a brief review of the same. You will find interesting articles on Map Reduce technology, monetary aspect of Open Source along with others and as usual a useful HowTo section.

We sincerely hope that you will like the issue. The magazine is a community effort and any interest in contribution (editing, layout, etc) is warmly welcome. Please send in your articles and feedback at theinitmag@gmail.com and help us spread the words.

Cheers,
The init Magazine Editorial Team

CONTENTS

Layout and Design: Abhishek Singh

Divide and Rule

Akash Deep Shakya
akashakya@gmail.com

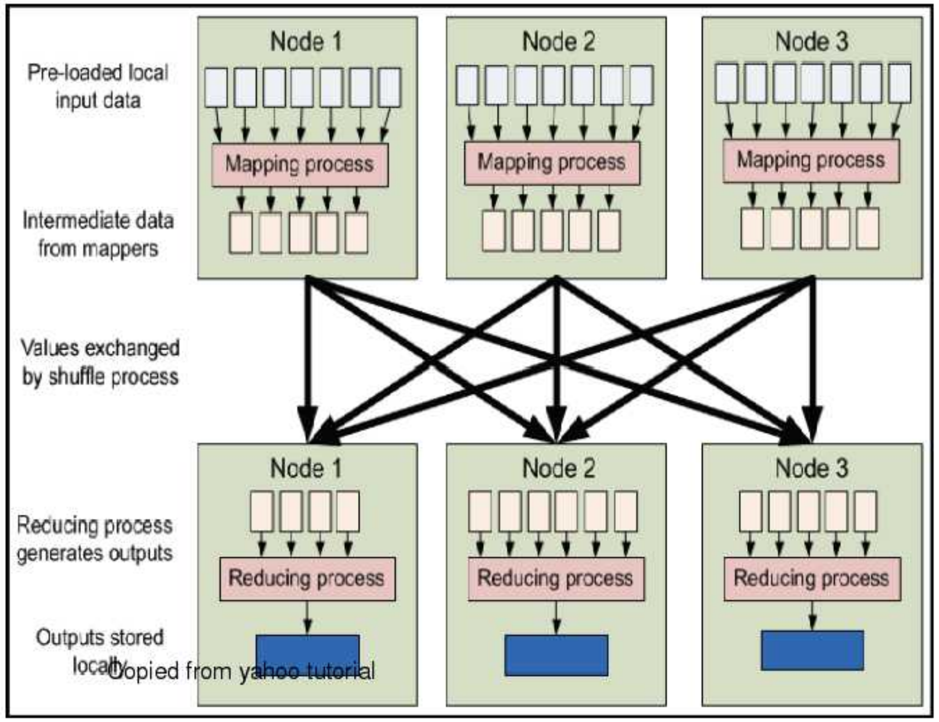
The world we live in is so big, thousand of millions of people live in here, they are busy, they do numerous activities and the most important thing for me is they do it on their computers. So when that happens, terabytes of data is generated every hour. If you think I am just fooling you around, let's take an example, the most important thing that a person presumes when he wakes up in the morning is not brushing his teeth or reading magazines along with his early morning restroom stuff, but it is updating his/her twitter status and checking who uploaded last night party's picture this morning. Then s/he starts liking/commenting these pictures, status etc. Just imagine, how facebook or twitter will store this data, manage the log files, replicate it for data backup; it's just too much of data. Due to this huge amount of data, the problems regarding the storage, performance, analysis and adhoc-interactive data representation started to show off.

We did it!! We again proved that human is the most intelligent animal in this earth, Google, yahoo, Microsoft already predicted that the data will be massive in future, so they worked for technology to handle such problem. For example, Google released the papers of their big table, google file system (GFS) and map reduce framework. Big table is a huge column orientated database for most of Google's applications, GFS is their own File System, and Map reduce is the framework which is used for most of the work they do, searching, indexing etc. With the help of Google's papers, Open Source Community worked on these technology to develop technology such as Hadoop framework, and on top of that created subprojects such as Hbase, CHukwa, ZooKeeper, Hive, Pig and Others. Along with Hadoop and it's subproject where major contribution comes from Apache Software Foundation, Stumble Upon, Yahoo, Microsoft (can you believe it!!) , Streamy and other; other technology such as Cassandra which is based on Amazon's Dynamo is also very popular. It is obviously very hard to incorporate every details in this article, however I would like to make a series of article where I will writing about Map Reduce, Hadoop, Cassandra, HBase and Hive; if possible Zooper and Chukwa as well. For now I will provide some references which you help you guys to read.

Map Reduce Framework

When I first tried to read this paper from Google, I felt like it is a rocket science. However, I tried to read it six to seven times without any success, so I tried to watch some of the videos of Map Reduce. And later I began to realize that, I did this thing when I was in grade 5, we used to play this game of organization all our words that we used to write in our Nepali text book and we used to create a dictionary out of it in the end. So my ambition is to save you time, and at the same time make your understanding of Map Reduce easier.

Let me assume that I am Google and I have entire web on my data centers. My data center is composed of millions of commodity hardware among then my three favorite servers are Foss (Node 1 in fig.1), Nepal (Node 2 in fig.1) and community (Node 3 in fig.1). These three servers contain the data of English Premier League


Node 1
Node 2
Node 3
Pre-loaded local input data
Mapping process
Intermediate data from mappers
Values exchanged by shuffle process
Node 1
Node 2
Node 3
Reducing process generates outputs
Reducing process
Outputs stored locally
Copied from yahoo tutorial

Source: Yahoo

season 2009. Now what I would like to know is who scored the highest number of goals in EPL 2009. We will use map reduce to find it out.

Considering the program definition, what we need to find out is players scoring goals. So what our Mapper task does is, it will search through the whole data (preloaded local input data in fig 1) and find out the players who scored goal, for example is Wayne Rooney have scored a goal in a match against West Ham United (Match 1 of the season) then our program will create a map, <Rooney, 1>. This means Rooney score 1 goal against West Ham United. So the output (Intermediate data from mappers) of the mapping process is the key value pair where we have key as the name of the player and the value the goal he scored. So we will create such key value pair for all the players who scored goals.

Note that, the data of Match 1-9 is in Foss server, 10-25 is in Nepal server and 26-38 is in Community server. This means, if Rooney scored 3 goals from Match 1-9, and he scored 5 goals in match 10-25 then we must communicate between two servers to get his goals from match 1-25, so what we do is, we write a reduce function. A reduce function will check each one of the players (keys) and try to add the number of goals they score. They take keys from all three servers and do the manipulation which gives the results as <Drogba, 29> turns out Didier Drogba is the one who scored the most goals.

This is how Map Reduce works. I am not sure how much you guys understood, but I am sure you had a pretty good knowledge of EPL now. Please do give me your feedbacks and for help email me at akashakya@gmail.com. I will catch up in next session of init magazine where I will write about Hadoop.

“When I first tried to read this paper from Google, I felt like it is a rocket science. However, I tried to read it six to seven times without any success, so I tried to watch some of the videos of Map Reduce. And later I began to realize that, I did this thing when I was in grade 5, we used to play this game of organization all our words that we used to write in our Nepali text book and we used to create a dictionary out of it in the end.”

FOSS

Where is money?

Biswas Parajuli
b@p.com

When dark clouds of "business-unfriendly" adjectives like 'free' and 'open-source' hover over our heads, there can be a drenching shower of some cold questions.

- How could a FOSS programmer make a living?
- If everything is available for free, who will do the necessary brainstorming for innovation?
- Doesn't it kill creativity?
- Is FOSS just a donation driven social work without any sense of profit making?

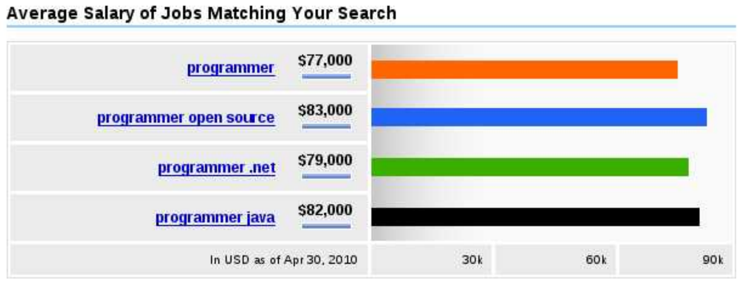
The doubts look plausible when we fail to relate it to the 'free' as in 'freedom' analogy, and if we do, still money seems to be missing. If we have a look at things that are happening around us :

"*Singapore actively plans to short circuit dependence by providing tax breaks to companies that use FOSS rather than proprietary software which is a macroeconomic decision by the government to foster the development of its own domestic industry in a certain, planned direction. It is also a strategic political decision related to a desire for political autonomy.*"(Deek and McHugh 2008, p. 313)

"*Nokia, which still rules the mobile roost launches its first open source Symbian phone*" (BBC NEWS, 27 Apr 2010)

Indeed.com, a job listing site shows "Av*erage programmer open source salaries for job postings nationwide are 8% higher than average programmer salaries for job postings nationwide.*" on the query http://www.indeed.com/salary?q1=programmer&l1=&q2=programmer+open+source&l2=&q3=programmer+.net&l3=&q4=programmer+java&l4= (see the graph)

So, with these we can atleast be sure that there is definitely some element of money involved. To understand the FOSS economics, first we need to see how it is made.



With the large number of programmers and developers spread all over the world, the conventional closed method of software development can not harness all the talents available. So, for FOSS development, the source code is exposed in the open which is then further enhanced by the interested coders spread world wide. The "interest" factors can be either academic project requirements or enthusiasm to learn by 'dirtying the hands' or funding by research institutes or plain hobby. Thus, the main principle and practice of open source software development is peer production by bartering and collaboration, with the end-product (and source-material) available at no cost to the public.

There is already some hidden "cash" involved inside.

Inherent Economics of FOSS :

A) Sharing and Bartering

In the past, when the concept of money was not invented yet, transactions were made by exchanging goods after relative evaluation. Even today, this Barter System retains its relevance. One main requirement of this system is that the transaction time should be nil. Software transaction fulfils this requirement since it is done through high speed internet. Suppose, Sushil makes a tool ‘aX’ and Mitesh makes another called ‘bY’ as per their needs. Now, their needs change and want each others tools. What they do is just exchange and share. A transaction has occured and thus, without any need of money an economic activity has been achieved.

B) Money Saved is Money Earned

When the source code is open and easily available, chances of ‘reinventing the wheel’ are greatly reduced. A function module already developed can be integrated into other projects. This saves the possible investment that would have been made in case the closed methodology was used. Also use of FOSS is less expensive than the use of proprietary software. The maintenance and upgrades are also cheaper or almost free.

There are many ways FOSS carries out its businesses and earns nice profit margins. Direct software sales as commodity as done by the conventional commercial software companies is not the spirit of the FOSS model.

In his paper, “The Magic Cauldron”, Eric S. Raymond analyzes the evolving economic substrate of the open-source phenomenon. He presents a game-theory analysis of the stability of open-source cooperation along with some profit seeking models for sustainable funding of open-source development.

FOSS BUSINESS MODELS :

The new Open Source Business Models basically deal with maximising use value, building a user community and creating Common Platforms.

1) Market positioner Model :

An open-source software is used to create or maintain a market position for proprietary software that generates a direct revenue stream. Netscape Communications, Inc. was pursuing this strategy when it open-sourced the Mozilla browser in early 1998 as its revenues started dropping when Microsoft first shipped Internet Explorer.

2) Widget Frosting Model:

In this model, a hardware company goes open-source in order to get better drivers and interface tools cheaper. Silicon Graphics, for example, supports and ships Samba.For widget-makers (such as semiconductor or peripheral-card manufacturers), interface software is not even potentially a revenue source. Therefore the downside of moving to open source is minimal.

3) Support Sellers Model :

It follows the “Give Away the Recipe, Open A Restaurant” formula. This model is used to create market for services by open sourcing the software. One can(effectively) give away the software product, but sell distribution, branding, and after-sale service. This is what Red Hat does.

4) Accessorizing :

In this model, accessories for open-source software are sold. At the low end, mugs and T-shirts; at the high end, professionally-edited and produced documentation like tutorials. Learning from the present craze for Microsoft certified courses, courses related to FOSS/GNU/Linux can also be marketed well.

5) Dual Licensing model:
Two versions of the software are maintained - FOSS version and Proprietary version. The proprietary version has a larger feature set with more testing and certification and is licensed for support revenues. MySQL and ReiserFS followed this model.

These models are constantly changing and evolving. Many years have been spent on analyzing and experimenting with open-source business models. As Tim O‘Reilly writes in “The Open Source Paradigm Shift” (2004), our understanding of open source is making a shift. FOSS is more of a business enabler than a business model.
We may not see the making of any new software giant like Microsoft in the future but what we can see is a world with a distributed framework of software development and revenue generation owing to the FOSS trends. Open source has thrived throughout the recession, providing low-cost, high-value solutions that appeal to firms faced with tight budgets.With the world economy returning to normalcy , the time is ripe to re-examine plans for technology deployments as many companies start investing in new IT projects.

So, the dark clouds are awaiting to be replaced with the warm, shining sun.

Resources :

www.apdip.net/publications/fosseprimers/foss-gov.pdf
http://www.catb.org/~esr/writings/magic-cauldron/magic-cauldron.html
http://news.bbc.co.uk/2/hi/8646715.stm
http://p2pfoundation.net/Open_Source_Business_Models
http://www.indeed.com/salary?q1=programmer&l1=&q2=programmer+open+source&l2=&q3=programmer+.net&l3=&q4=programmer+java&l4=
http://atulchitnis.net/talks/nitc-biz.pdf
http://tim.oreilly.com/articles/paradigmshift_0504.html