From: Bill Gates
Sent: Wednesday, November 04, 1998 11:03 AM
To: Bob Herbold
Cc: Steve Ballmer
Subject: FW: Open Source Software & Linux

The two documents in here from Vinod are the ones I want the board to see.

—Original Message—
From: Vinod Valloppillil (Exchange)
Sent: Wednesday, November 04, 1998 11:01 AM
To: Bill Gates; Eric Rudder
Subject: FW: Open Source Software & Linux

here you go.

—Original Message—
From: Vinod Valloppillil (Exchange)
Sent: Tuesday, August 11, 1998 7:08 PM
To: Bill Gates; Steve Ballmer; Paul Maritz; Jim Allchin (Exchange); Jeff Raikes; Rick Rashid; Nathan Myhrvold; Craig Mundie; Bob Muglia (Exchange); Jim Allchin's Direct Reports; Brad Chase; Brad Chase's Direct Reports; Rich Tong; Moshe Dunie; Moshe Dunie's Direct Reports; David Vaskevitch; Tod Nielsen
Cc: Vinod Valloppillil (Exchange); Josh Cohen
Subject: Open Source Software & Linux

*** Microsoft Confidential ***

I've authored a pair of documents analyzing **Open Source Software (OSS)** and the **Linux Operating System**. Because these topics challenge many of our assumptions about the economics of software, competition, and development techniques, these docs are a top-to-bottom immersion in their lingo and processes.

Recently, **Open Source Software (OSS)** has garnered significant attention from developers, customers, trade and national media who herald it as a new form of software development and as an alternative to the "demons" of commercial software. The deep interdependence of today's OSS projects and the Internet in many ways makes OSS the first "internet-native" development system.

Executive Summary:

Open Source Software (OSS) is a development process which promotes rapid creation and deployment of incremental features and bug fixes into an existing code / knowledge base. In recent years, corresponding to the growth of Internet, OSS projects have acquired the depth & complexity traditionally associated with commercial projects such as Operating Systems and mission critical servers.

Consequently, OSS poses a direct, short-term revenue and platform threat to Microsoft - particularly in the server space. Additionally, the intrinsic parallelism and free idea exchange in OSS has benefits that are not replicable with our current licensing model and therefore present a long term developer mindshare threat.

oss.doc (244 KB)

The most widely recognized and significant OSS project is the **Linux Operating System**. The second document, co-authored with Josh Cohen, analyzes Linux and its strategic impact on Windows.

Executive Summary:

The Linux OS is the highest visibility product of the Open Source Software (OSS) process. Linux represents a best-of-breed UNIX, that is trusted in mission critical applications, and - due to it's open source code - is more long term credible than other competitive OS's.

Linux poses a significant near-term revenue threat to Windows NT Server in the commodity file, print and network services businesses. Linux's emphasis on serving the hacker and UNIX community alleviates the near-medium term potential for damage to the Windows client desktop.

linux.doc (481 KB)

Both docs contain some initial ideas which are a starting point for discussing a well-formulated, cohesive response to this phenomena. OSS presents Microsoft not only with a new threat but also a new opportunity to reach the next, wider circle of developers & enrich the Windows platform and applications.

- VinodV

Microsoft Corporation

Open Source Software
A (New?) Development Methodology

Vinod Valloppillil (VinodV)
Aug 11, 1998 – v1.00

Microsoft Confidential

Open Source [redacted]oftware
A (New?) De[redacted]opment Methodology
Executive Summary

Open Source Software (OSS) is a development process which promotes rapid creation and deployment of incremental features and bug fixes in an existing code / knowledge base. In recent years, corresponding to the growth of Internet, OSS projects have acquired the depth & complexity traditionally associated with commercial projects such as Operating Systems and mission critical servers.

Consequently, OSS poses a direct, short-term revenue and platform threat to Microsoft – particularly in server space. Additionally, the intrinsic parallelism and free idea exchange in OSS has benefits that are not replicable with our current licensing model and therefore present a long term developer mindshare threat.

However, other OSS process weaknesses provide an avenue for Microsoft to garner advantage in key feature areas such as architectural improvements (e.g. storage+), integration (e.g. schemas), ease-of-use, and organizational support.

Open Source Software
What is it?

Open Source Software (OSS) is software in which both source and binaries are distributed or accessible for a given product, usually for free. OSS is often mistaken for "shareware" or "freeware" but there are significant differences between these licensing models and the process around each product.

Software Licensing Taxonomy

Software Type							
Commercial							
Trial Software	X (Non-full featured)	X					
Non-Commercial Use	X (Usage dependent)	X					
Shareware	X- (Unenforced licensing)	X					
Royalty-free binaries ("Freeware")	X	X	X				
Royalty-free libraries	X	X	X	X			
Open Source (BSD-Style)	X	X	X	X	X		
Open Source (Apache Style)	X	X	X	X	X	X	
Open Source (Linux/GNU style)	X	X	X	X	X	X	X
License Feature	**Zero Price Avenue**	**Redistributable**	**Unlimited Usage**	**Source Code Available**	**Source Code Modifiable**	**Public "Check-ins" to core codebase**	**All derivatives must be free**

The broad categories of licensing include:

- **Commercial software**
  Commercial software is classic Microsoft bread-and-butter. It must be purchased, may NOT be redistributed, and is typically only available as binaries to end users.
- **Limited trial software**

Limited trial software are usually functionally limited versions of commercial software which are freely distributed and intend to drive purchase of the commercial code. Examples include 60-day time bombed evaluation products.

- **Shareware**
  Shareware products are fully functional and freely redistributable but have a license that mandates eventual purchase by both individuals and corporations. Many internet utilities (like "WinZip") take advantage of shareware as a distribution method.
- **Non-commercial use**
  Non-commercial use software is freely available and redistributable by non-profit making entities. Corporations, etc. must purchase the product. An example of this would be Netscape Navigator.
- **Royalty free binaries**
  Royalty-free binaries consist of software which may be freely used and distributed in binary form only. Internet Explorer and NetMeeting binaries fit this model.
- **Royalty free libraries**
  Royalty-free libraries are software products whose binaries and source code are freely used and distributed but may NOT be modified by the end customer without violating the license. Examples of this include class libraries, header files, etc.
- **Open Source (BSD-style)**
  A small, closed team of developers develops BSD-style open source products & allows free use and redistribution of binaries and code. While users are allowed to modify the code, the development team does NOT typically take "check-ins" from the public.
- **Open Source (Apache-style)**
  Apache takes the BSD-style open source model and extends it by allowing check-ins to the core codebase by external parties.
- **Open Source (CopyLeft, Linux-style)**
  CopyLeft or GPL (General Public License) based software takes the Open Source license one critical step farther. Whereas BSD and Apache style software permits users to "fork" the codebase and apply their own license terms to their modified code (e.g. make it commercial), the GPL license requires that all derivative works in turn must also be GPL code. "You are free to hack this code as long as your derivative is also hackable"

Open Source Software is Significant to Microsoft

This paper focuses on Open Source Software (OSS). OSS is acutely different from the other forms of licensing (in particular "shareware") in two very important respects:

1. There always exists an avenue for completely royalty-free purchase of the core code base
2. Unlike freely distributed binaries, Open Source encourages a *process* around a core code base and encourages extensions to the codebase by other developers.

OSS is a concern to Microsoft for several reasons:

1. **OSS projects have achieved "commercial quality"**
   A key barrier to entry for OSS in many customer environments has been its perceived lack of quality. OSS advocates contend that the greater code inspection & debugging in OSS software results in higher quality code than commercial software.
   Recent case studies (the Internet) provide very dramatic evidence in customer's eyes that commercial quality can be achieved / exceeded by OSS projects. At this time, however there is no strong evidence of OSS code quality aside from anecdotal.
2. **OSS projects have become large-scale & complex**
   Another barrier to entry that has been tackled by OSS is project complexity. OSS teams are undertaking projects whose size & complexity had heretofore been the exclusive domain of commercial, economically-organized/motivated development teams.
   Examples include the Linux Operating System and Xfree86 GUI.
   OSS process vitality is directly tied to the Internet to provide distributed development resources on a mammoth scale. Some examples of OSS project size:

Project	Lines of Code
Linux Kernel (x86 only)	500,000
Apache Web Server	80,000
SendMail	57,000
Xfree86 X-windows server	1.5 Million
"K" desktop environment	90,000
Full Linux distribution	~10 Million

3. **OSS has a unique development process with unique strengths/weaknesses**
   The OSS process is unique in its participants' motivations and the resources that can be brought to bare down on problems. OSS, therefore, has some interesting, non-replicable assets which should be thoroughly understood.

History

Open source software has roots in the hobbyist and the scientific community and was typified by ad hoc exchange of source code by developers/users

Internet Software

The largest case study of OSS is the Internet. Most of the earliest code on the Internet was, and is still based on OSS as described in an interview with Tim O'Reilly (http://www.techweb.com/internet/profile/toreilly/interview):

> *TIM O'REILLY: The biggest message that we started out with was, "open source software works." ... BIND has absolutely dominant market share as the single most mission-critical piece of software on the Internet. Apache is the dominant Web server. SendMail runs probably eighty percent of the mail servers and probably touches every single piece of e-mail on the Internet*

Free Software Foundation / GNU Project

Credit for the first instance of modern, organized OSS is generally given to Richard Stallman of MIT. In late 1983, Stallman created the Free Software Foundation (FSF) – http://www.gnu.ai.mit.edu/fsf/fsf.html -- with the goal of creating a free version of the UNIX operating system. The FSF released a series of sources and binaries under the GNU moniker (which recursively stands for "Gnu's Not Unix").

The original FSF / GNU initiatives fell short of their original goal of creating a completely OSS Unix1. They did, however, contribute several famous and widely disseminated applications and programming tools used today including:

- **GNU Emacs** – originally a powerful character-mode text editor, over time Emacs was enhanced to provide a front-end to compilers, mail readers, etc.
- **GNU C Compiler (GCC)** – GCC is the most widely used compiler in academia & the OSS world. In addition to the compiler a fairly standardized set of intermediate libraries are available as a superset to the ANSI C libraries.
- **GNU GhostScript** – Postscript printer/viewer.

CopyLeft Licensing

FSF/GNU software introduced the "copyleft" licensing scheme that not only made it illegal to hide source code from GNU software but also made it illegal to hide the source from work *derived from GNU software*. The document that described this license is known as the General Public License (GPL). Wired magazine has the following summary of this scheme & its intent (http://www.wired.com/wired/5.08/linux.html):

> *The general public license, or GPL, allows users to sell, copy, and change copylefted programs - which can also be copyrighted - but you must pass along the same freedom to sell or copy your modifications and change them further. You must also make the source code of your modifications freely available.*

---

1 FSF failed because:

- Stallman was often accused of Micro-Management
- Stallman required that all code in GNU be GPL'd whereas Linux was willing to live with the LGPL (described later.)

The second c[redacted]se – open source code of derivative works – has been the most controversial (and, potentially th[redacted]ost successful) aspect of CopyLeft licensing.

Open Source Process

Commercial software development processes are hallmarked by organization around economic goals. However, since money is often not the (primary) motivation behind Open Source Software, understanding the nature of the threat posed requires a deep understanding of the process and motivation of Open Source development teams

In other words, to understand how to compete against OSS, we must target a process rather than a company.

Open Source Development Teams

Some of the key attributes of Internet-driven OSS teams:

- Geographically far-flung. Some of the key developers of Linux, for example, are uniformly distributed across Europe, the US, and Asia.
- Large set of contributors with a smaller set of core individuals. Linux, once again, has had over 1000 people submit patches, bug fixes, etc. and has had over 200 individuals directly contribute code to the kernel.
- Not monetarily motivated (in the short run). These individuals are more like hobbyists spending their free time / energy on OSS project development while maintaining other full time jobs. This has begun to change somewhat as commercial versions of the Linux OS have appeared.

OSS Development Coordination

Communication – Internet Scale

Coordination of an OSS team is extremely dependent on Internet-native forms of collaboration. Typical methods employed run the full gamut of the Internet's collaborative technologies:

- Email lists
- Newsgroups
  - 24x 7 monitoring by international subscribers
- Web sites

OSS projects the size of Linux and Apache are only viable if a large enough community of highly skilled developers can be amassed to attack a problem. Consequently, there is direct correlation between the size of the project that OSS can tackle and the growth of the Internet.

Common Direction

In addition to the communications medium, another set of factors implicitly coordinate the direction of the team.

Common Goals

Common goals are the equivalent of vision statements which permeate the distributed decision making for the entire development team. A single, clear directive (e.g. "recreate UNIX") is far more efficiently communicated and acted upon by a group than multiple, intangible ones (e g. "make a good operating system").

Common Precedents

Precedence is potentially the most important factor in explaining the rapid and cohesive growth of massive OSS projects such as the Linux Operating System. Because the entire Linux community has years of shared experience dealing with many other forms of UNIX, they are easily able to discern – in a non-confrontational manner – what worked and what didn't.

There weren't arguments about the command syntax to use in the text editor – everyone already used "vi" and the developers simply parcelled out chunks of the command namespace to develop.

Having historical, 20:20 hindsight provides a strong, implicit structure. In more forward looking organizations, this structure is provided by strong, visionary leadership.

Common Skillsets

NatBro points out that the need for a commonly accepted skillset as a pre-requisite for OSS development. This point is closely related to the common precedents phenomena. From his email:

> A key attribute … is the common UNIX/gnu/make skillset that OSS taps into and

reinforces. I think the whole process wouldn't work if the barrier to entry were much higher than it is .. a modestly skilled UNIX programmer can grow into doing great things with Linux and many OSS products2. Put another way -- it's not too hard for a developer in the OSS space to scratch their itch, because things build very similarly to one another, debug similarly, etc.

Whereas precedents identify the end goal, the common skillsets attribute describes the number of people who are versed in the process necessary to reach that end.

The Cathedral and the Bazaar

A very influential paper by an open source software advocate – Eric Raymond – was first published in May 1997 (http://www.redhat.com/redhat/cathedral-bazaar/). Raymond's paper was expressly cited by (then) Netscape CTO Eric Hahn as a motivation for their decision to release browser source code. Raymond dissected his OSS project in order to derive rules-of-thumb which could be exploited by other OSS projects in the future. Some of Raymond's rules include:

Every good work of software starts by scratching a developer's personal itch

This summarizes one of the core motivations of developers in the OSS process – solving an immediate problem at hand faced by an individual developer – this has allowed OSS to evolve complex projects without constant feedback from a marketing / support organization.

Good programmers know what to write. Great ones know what to rewrite (and reuse).

Raymond posits that developers are more likely to reuse code in a rigorous open source process than in a more traditional development environment because they are always guaranteed access to the entire source all the time.

Widely available open source reduces search costs for finding a particular code snippet.

``Plan to throw one away; you will, anyhow."

Quoting Fred Brooks, ``The Mythical Man-Month", Chapter 11. Because development teams in OSS are often extremely far flung, many major subcomponents in Linux had several initial prototypes followed by the selection and refinement of a single design by Linus

Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging

Raymond advocates strong documentation and significant developer support for OSS projects in order to maximize their benefits

Code documentation is cited as an area which commercial developers typically neglect which would be a fatal mistake in OSS.

Release early. Release often. And listen to your customers.

This is a classic play out of the Microsoft handbook. OSS advocates will note, however, that their release-feedback cycle is potentially an order of magnitude faster than commercial software's.

Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone.

This is probably the heart of Raymond's insight into the OSS process. He paraphrased this rule as "debugging is parallelizable". More in depth analysis follows.

Parallel Development

Once a component3 framework has been established (e.g. key API's & structures defined), OSS projects such as Linux utilize multiple small teams of individuals independently solving particular problems. Because the developers are typically hobbyists, the ability to 'fund' multiple, competing efforts is not an issue and the OSS process benefits from the ability to pick the best potential implementation out of the

---

2 A great account of a win32 developer who "discovered" the Linux world clearly describes his "feeling" as he "grew" into becoming a better OSS developer: http://www.osnews.com/features/08.98/view.html. I liked this article because, unlike the religious zealotry on other OSS web pages, this account is very factual and metered.

3 By "component" I'm referring to source code modules / functions rather than binary components in the COM sense.

many produc[redacted]
Note, that thi[redacted]s very dependent on:

- A large group of individuals willing to submit code
- A strong, implicit componentization framework (which, in the case of Linux was inherited from UNIX architecture).

Parallel Debugging
The core argument advanced by Eric Raymond is that unlike other aspects of software development, code debugging is an activity whose efficiency improves nearly linearly with the number of individuals tasked with the project. There are little/no management or coordination costs associated with debugging a piece of open source code – this is the key 'break' in Brooks' laws for OSS.
Raymond includes Linus Torvald's description of the Linux debugging process:
My original formulation was that every problem ``will be transparent to somebody''. Linus demurred that the person who understands and fixes the problem is not necessarily or even usually the person who first characterizes it. ``Somebody finds the problem,'' he says, ``and somebody else understands it. And I'll go on record as saying that finding it is the bigger challenge.'' But the point is that both things tend to happen quickly

Put alternately:
``Debugging is parallelizable''. Jeff [Dutky <dutky@wam.umd.edu>] observes that although debugging requires debuggers to communicate with some coordinating developer, it doesn't require significant coordination between debuggers. Thus it doesn't fall prey to the same quadratic complexity and management costs that make adding developers problematic.

> One advantage of parallel debugging is that bugs and their fixes are found / propagated much faster than in traditional processes. For example, when the TearDrop IP attack was first posted to the web, less than 24 hours passed before the Linux community had a working fix available for download.

"Impulse Debugging"
An extension to parallel debugging that I'll add to Raymond's hypothesis is "impulsive debugging". In the case of the Linux OS, implicit to the act of installing the OS is the act of installing *the debugging/development environment*. Consequently, it's highly likely that if a particular user/developer comes across a bug in another individual's component – and especially if that bug is "shallow" – that user can very quickly patch the code and, via internet collaboration technologies, propagate that patch very quickly back to the code maintainer.
Put another way, OSS processes have a very low entry barrier to the debugging process due to the common development/debugging methodology derived from the GNU tools.
Conflict resolution
Any large scale development process will encounter conflicts which must be resolved. Often resolution is an arbitrary decision in order to further progress the project. In commercial teams, the corporate hierarchy + performance review structure solves this problem – How do OSS teams resolve them?
In the case of Linux, Linus Torvalds is the undisputed 'leader' of the project. He's delegated large components (e.g. networking, device drivers, etc.) to several of his trusted "lieutenants' who further de-facto delegate to a handful of "area" owners (e.g. LAN drivers).
Other organizations are described by Eric Raymond:
(http://earthspace.net/~esr/writings/homesteading/homesteading-15.html):
Some very large projects discard the 'benevolent dictator' model entirely. One way to do this is turn the co-developers into a voting committee (as with Apache) Another is rotating dictatorship, in which control is occasionally passed from one member to another within a circle of senior co-developers (the Perl developers organize themselves this way).

Motivation

This section provides an overview of some of the key reasons OSS developers seek to contribute to OSS projects.

Solving the Problem at Hand

This is basically a rephrasing of Raymond's first rule of thumb – "Every good work of software starts by scratching a developer's personal itch".

Many OSS projects – such as Apache – started as a small team of developers setting out to solve an immediate problem at hand. Subsequent improvements of the code often stem from individuals applying the code to their own scenarios (e.g. discovering that there is no device driver for a particular NIC, etc.)

Education

The Linux kernel grew out of an educational project at the University of Helsinki. Similarly, many of the components of Linux / GNU system (X windows GUI, shell utilities, clustering, networking, etc.) were extended by individuals at educational institutions.

- In the Far East, for example, Linux is reportedly growing faster than internet connectivity – due primarily to educational adoption.
- Universities are some of the original proponents of OSS as a teaching tool.
- Research/teaching projects on top of Linux are easily 'disseminated' due to the wide availability of Linux source. In particular, this often means that new research ideas are first implemented and available on Linux before they are available / incorporated into other platforms.

Ego Gratification

The most ethereal, and perhaps most profound motivation presented by the OSS development community is pure ego gratification.

In "The Cathedral and the Bazaar", Eric S. Raymond cites.

> The ``utility function" Linux hackers are maximizing is not classically economic, but is the intangible of their own ego satisfaction and reputation among other hackers.

And, of course, "you aren't a hacker until someone else calls you hacker"

Homesteading on the Noosphere

A second paper published by Raymond – "Homesteading on the Noosphere" (http://sagan.earthspace.net/~esr/writings/homesteading/), discusses the difference between economically motivated exchange (e.g. commercial software development for money) and "gift exchange" (e.g. OSS for glory).

"Homesteading" is acquiring property by being the first to 'discover' it or by being the most recent to make a significant contribution to it. The "Noosphere" is loosely defined as the "space of all work". Therefore, Raymond posits, the OSS hacker motivation is to lay a claim to the largest area in the body of work. In other words, take credit for the biggest piece of the prize.

From "Homesteading on the Noosphere":

> Abundance makes command relationships difficult to sustain and exchange relationships an almost pointless game. In gift cultures, social status is determined not by what you control but by what you give away
> ...

For examined in this way, it is quite clear that the society of open-source hackers is in fact a gift culture. Within it, there is no serious shortage of the `survival necessities' – disk space, network bandwidth, computing power. Software is freely shared. This abundance creates a situation in which the only available measure of competitive success is reputation among one's peers.

More succinctly (http://www techweb.com/internet/profile/eraymond/interview):

> SIMS: So the scarcity that you looked for was the scarcity of attention and reward?
> RAYMOND: That's exactly correct.

Altruism

This is a controversial motivation and I'm inclined to believe that at some level, Altruism 'degenerates' into a form of the Ego Gratification argument advanced by Raymond.

Open Source [redacted]ftware – 08/01/03; 4:19 PM

One smaller m[redacted]tivation which, in part, stems from altruism is Microsoft-bashing.

Code Forking

A key threat i[redacted]any large development team – and one that is particularly exacerbated by the process chaos of an internet-scale development team -- is the risk of code-forking.

Code forking occurs when over normal push-and-pull of a development project, multiple, inconsistent versions of the project's code base evolve.

In the commercial world, for example, the strong, singular management of the Windows NT codebase is considered to be one of it's greatest advantages over the 'forked' codebase found in commercial UNIX implementations (SCO, Solaris, IRIX, HP-UX, etc.).

Forking in OSS – BSD Unix

Within OSS space, BSD Unix is the best example of forked code. The original BSD UNIX was an attempt by U-Cal Berkeley to create a royalty-free version of the UNIX operating system for teaching purposes. However, Berkeley put severe restrictions on non-academic uses of the codebase.

In order to create a fully free version of BSD UNIX, an ad hoc (but closed) team of developers created FreeBSD. Other developers at odds with the FreeBSD team for one reason or another splintered the OS to create other variations (OpenBSD, NetBSD, BSDI).

There are two dominant factors which led to the forking of the BSD tree:

- Not everyone can contribute to the BSD codebase. This limits the size of the effective "Noosphere" and creates the potential for someone else to credibly claim that their forked code will become more dominant than the core BSD code.
- Unlike GPL, BSD's license places no restrictions on derivative code. Therefore, if you think your modifications are cool enough, you are free to fork the code, charge money for it, change its name, etc.

Both of these motivations create a situation where developers may try to force a fork in the code and collect royalties (monetary, or ego) at the expense of the collective BSD society.

(Lack of) Forking in Linux

In contrast to the BSD example, the Linux kernel code base hasn't forked. Some of the reasons why the integrity of the Linux codebase has been maintained include:

- Universally accepted leadership
  Linus Torvalds is a celebrity in the Linux world and his decisions are considered final. By contrast, a similar celebrity leader did NOT exist for the BSD-derived efforts.
  Linus is considered by the development team to be a fair, well-reasoned code manager and his reputation within the Linux community is quite strong. However, Linus doesn't get involved in every decision. Often, sub groups resolve their – often large – differences amongst themselves and prevent code forking.
- Open membership & long term contribution potential.
  In contrast to BSD's closed membership, anyone can contribute to Linux and your "status" – and therefore ability to 'homestead' a bigger piece of Linux – is based on the size of your previous contributions.
  Indirectly this presents a further disincentive to code forking. There is almost no credible mechanism by which the forked, minority code base will be able to maintain the rate of innovation of the primary Linux codebase.
- GPL licensing eliminates economic motivations for code forking
  Because derivatives of Linux MUST be available through some free avenue, it lowers the long term economic gain for a minority party with a forked Linux tree.
- Forking the codebase also forks the "Noosphere"
  Ego motivations push OSS developers to plant the biggest stake in the biggest Noosphere. Forking the code base inevitably shrinks the space of accomplishment for any subsequent developers to the new code tree.

Open Source Strengths

What are the core strengths of OSS products that Microsoft needs to be concerned with?

OSS Exponential Attributes

Like our Operating System business, OSS ecosystems have several exponential attributes:

- **OSS processes are growing with the Internet**
  The single biggest constraint faced by any OSS project is finding enough developers interested in contributing their time towards the project. As an enabler, the Internet was absolutely necessary to bring together enough people for an Operating System scale project. More importantly, the *growth engine* for these projects is the growth in the Internet's reach. Improvements in collaboration technologies directly lubricate the OSS engine
  Put another way, the growth of the Internet will make existing OSS projects bigger and will make OSS projects in "smaller" software categories become viable.
- **OSS processes are "winner-take-all"**
  Like commercial software, the most viable single OSS project in many categories will, in the long run, kill competitive OSS projects and 'acquire' their IQ assets. For example, Linux is killing BSD Unix and has absorbed most of its core ideas (as well as ideas in the commercial UNIXes). This feature confers huge first mover advantages to a particular project
- **Developers seek to contribute to the largest OSS platform**
  The larger the OSS project, the greater the prestige associated with contributing a large, high quality component to its Noosphere. This phenomena contributes back to the "winner-take-all" nature of the OSS process in a given segment.
- **Larger OSS projects solve more "problems at hand"**
  The larger the project, the more development/test/debugging the code receives. The more debugging, the more people who deploy it.

Long-term credibility

Binaries may die but source code lives forever

One of the most interesting implications of viable OSS ecosystems is long-term credibility.

Long-Term Credibility Defined

Long term credibility exists if there is no way you can be driven out of business in the near term. This forces change in how competitors deal with you.

For example, Airbus Industries garnered initial long term credibility from explicit government support. Consequently, when bidding for an airline contract, Boeing would be more likely to accept short-term, non-economic returns when bidding against Lockheed than when bidding against Airbus.

Loosely applied to the vernacular of the software industry, a product/process is long-term credible if FUD tactics can not be used to combat it.

OSS is Long-Term Credible

OSS systems are considered credible because the source code is available from potentially millions of places and individuals.

The likelihood that Apache will cease to exist is orders of magnitudes lower than the likelihood that WordPerfect, for example, will disappear. The disappearance of Apache is not tied to the disappearance of binaries (which are affected by purchasing shifts, etc.) but rather to the disappearance of source code and the knowledge base.

Inversely stated, customers know that Apache will be around 5 years from now -- provided there exists some minimal sustained interested from its user/development community.

One Apache customer, in discussing his rationale for running his e-commerce site on OSS stated, "because it's open source, I can assign one or two developers to it and maintain it myself indefinitely 4"

Lack of Code-Forking Compounds Long-Term Credibility

The GPL and its aversion to code forking reassures customers that they aren't riding an evolutionary 'dead-end' by subscribing to a particular commercial version of Linux.

The "evolutionary dead-end" is the core of the software FUD argument.

---

4 Dwight Krossa pointed out in the early days of NT 3.1, some large customers asked to keep a copy of NT source code in exchange for long term contracts as a similar credibility guarantee.

Also, note that the second order, and larger, effect of the open source isn't that this developer can maintain the codebase indefinitely but rather, that as long as there are at least two developers somewhere on the net, the code can be indefinitely maintained

Parallel Debu[redacted]ging

Linux and ot[redacted] OSS advocates are making a progressively more credible argument that OSS software is at least as rob[redacted]t – if not more – than commercial alternatives. The Internet provides an ideal, high-visibility showcase for the OSS world.

In particular, larger, more savvy, organizations who rely on OSS for business operations (e.g. ISPs) are comforted by the fact that they can potentially fix a work-stopping bug independent of a commercial provider's schedule (for example, UUNET was able to obtain, compile, and apply the teardrop attack patch to their deployed Linux boxes within 24 hours of the first public attack)

Parallel Development

Alternatively stated, "developer resources are essentially free in OSS". Because the pool of potential developers is massive, it is economically viable to simultaneously investigate multiple solutions / versions to a problem and chose the best solution in the end.

For example, the Linux TCP/IP stack was probably rewritten 3 times. Assembly code components in particular have been continuously hand tuned and refined.

OSS = 'perfect' API evangelization / documentation

OSS's API evangelization / developer education is basically providing the developer with the underlying code. Whereas evangelization of API's in a closed source model basically defaults to trust, OSS API evangelization lets the developer make up his own mind.

NatBro and Ckindel point out a split in developer capabilities here. Whereas the "enthusiast developer" is comforted by OSS evangelization, novice/intermediate developers –the bulk of the development community – prefer the trust model + organizational credibility (e.g. "Microsoft says API X looks this way")

Release rate

Strongly componentized OSS projects are able to release subcomponents as soon as the developer has finished his code. Consequently, OSS projects rev quickly & frequently.

Open Source Weaknesses

The weaknesses in OSS projects fall into 3 primary buckets:

- Management costs
- Process Issues
- Organizational Credibility

Management Costs

The biggest roadblock for OSS projects is dealing with exponential growth of management costs as a project is scaled up in terms of rate of innovation and size. This implies a limit to the rate at which an OSS project can innovate.

Starting an OSS project is difficult

From Eric Raymond.

> It's fairly clear that one cannot code from the ground up in bazaar style. One can test, debug and improve in bazaar style, but it would be very hard to originate a project in bazaar mode. Linus didn't try it. I didn't either. Your nascent developer community needs to have something runnable and testable to play with.

Raymond 's argument can be extended to the difficulty in starting/sustaining a project if there are no clear precedent / goal (or too many goals) for the project.

Bazaar Credibility

Obviously, there are far more fragments of source code on the Internet than there are OSS communities. What separates "dead source code" from a thriving bazaar?

One article (http://www.mibsoftware.com/bazdev/0003.htm) provides the following *credibility* criteria:

> "....thinking in terms of a hard minimum number of participants is misleading. Fetchmail and Linux have huge numbers of beta testers *now*, but they obviously both had very few at the beginning.
>
> What both projects did have was a handful of enthusiasts and a plausible promise. The promise was partly technical (this code will be wonderful with a little effort) and sociological (if you join our gang, you'll have as much fun as we're having). So what's

necessary for a bazaar to develop is that it be credible that the full-blown bazaar will exist!"

I'll posit that some of the key criteria that must exist for a bazaar to be credible include:

- **Large Future Noosphere** – The project must be cool enough that the intellectual reward adequately compensates for the time invested by developers. The Linux OS excels in this respect.
- **Scratch a big itch** – The project must be important / deployable by a large audience of *developers*. The Apache web server provides an excellent example here.
- **Solve the right amount of the problem first** – Solving too much of the problem relegates the OSS development community to the role of testers. Solving too little before going OSS reduces "plausible promise" and doesn't provide a strong enough component framework to efficiently coordinate work.

Post-Parity Development

When describing this problem to JimAll, he provided the perfect analogy of "chasing tail lights". The easiest way to get coordinated behavior from a large, semi-organized mob is to point them at a known target5 Having the taillights provides concreteness to a fuzzy vision. In such situations, having a taillight to follow is a proxy for having strong central leadership.

Of course, once this implicit organizing principle is no longer available (once a project has achieved "parity" with the state-of-the-art), the level of management necessary to push towards new frontiers becomes massive.

This is possibly the single most interesting hurdle to face the Linux community now that they've achieved parity with the state of the art in UNIX in many respects.

Un-sexy work

Another interesting thing to observe in the near future of OSS is how well the team is able to tackle the "unsexy" work necessary to bring a commercial grade product to life.

In the operating systems space, this includes small, essential functions such as power management, suspend/resume, management infrastructure, UI niceties, deep Unicode support, etc.

For Apache, this may mean novice-administrator functionality such as wizards.

Integrative/Architectural work

Integrative work across modules is the biggest cost encountered by OSS teams. An email memo from Nathan Myrhvold on 5/98, points out that of all the aspects of software development, integration work is most subject to Brooks' laws.

Up till now, Linux has greatly benefited from the integration / componentization model pushed by previous UNIX's. Additionally, the organization of Apache was simplified by the relatively simple, fault tolerant specifications of the HTTP protocol and UNIX server application design.

Future innovations which require changes to the core architecture / integration model are going to be incredibly hard for the OSS team to absorb because it simultaneously devalues their precedents and skillsets.

Process Issues

These are weaknesses intrinsic to OSS's design/feedback methodology

Iterative Cost

One of the key's to the OSS process is having many more iterations than commercial software (Linux was known to rev it's kernel more than once a day!). However, commercial customers tell us they want fewer revs, not more.

"Non-expert" Feedback

The Linux OS is not developed for end users but rather, for other hackers. Similarly, the Apache web server is implicitly targetted at the largest, most savvy site operators, not the departmental intranet server

The key thread here is that because OSS doesn't have an explicit marketing / customer feedback component, wishlists – and consequently feature development – are dominated by the most technically savvy users.

One thing that development groups at MSFT have learned time and time again is that ease of use, UI intuitiveness, etc. must be built from the ground up into a product and can not be pasted on at a later time.

---

5 Microsoft development teams often function this way.

Open Source [redacted]ftware – 08/01/03; 4:19 PM

The interesting trend to observe here will be the effect that commercial OSS providers (such as RedHat in Linux space, [redacted]Net in Apache space) will have on the feedback cycle.

Organizational Credibility

How can OSS provide the *service* that consumers expect from software providers?

Support Model

Product support is typically the first issue prospective consumers of OSS packages worry about and is the primary feature that commercial redistributors tout.

However, the vast majority of OSS projects are supported by the developers of the respective components. Scaling this support infrastructure to the level expected in commercial products will be a significant challenge. There are many orders of magnitude difference between users and developers in IIS vs. Apache.

For the short-medium run, this factor alone will relegate OSS products to the top tiers of the user community.

Strategic Futures

A very sublime problem which will affect full scale consumer adoption of OSS projects is the lack of strategic direction in the OSS development cycle. While incremental improvement of the current bag of features in an OSS product is very credible, *future features* have no organizational commitment to guarantee their development.

What does it mean for the Linux community to "sign up" to help build the Corporate Digital Nervous System? How can Linux guarantee backward compatibility with apps written to previous API's? Who do you sue if the next version of Linux breaks some commitment? How does Linux make a strategic alliance with some other entity?

Open Source Business Models

In the last 2 years, OSS has taken another twist with the emergence of companies that sell OSS software, and more importantly, hiring full-time developers to improve the code base. What's the business model that justifies these salaries?

In many cases, the answers to these questions are similar to "why should I submit my protocol/app/API to a standards body?"

Secondary Services

The vendor of OSS-ware provides sales, support, and integration to the customer. Effectively, this transforms the OSS-ware vendor from a package goods manufacturer into a services provider.

Loss Leader -- Market Entry

The Loss Leader OSS business model can be used for two purposes:

- Jumpstarting an infant market
- Breaking into an existing market with entrenched, closed-source players

Many OSS startups – particularly those in Operating Systems space -- view funding the development of OSS products as a strategic loss leader against Microsoft.

Linux distributors, such as RedHat, Caldera, and others, are expressly willing to fund full time developers who release all their work to the OSS community. By simultaneously funding these efforts, Red Hat and Caldera are implicitly colluding and believe they'll make more short term revenue by growing the Linux market rather than directly competing with each other.

An indirect example is O'Reilly & Associates employment of Larry Wall – "leader" and full time developer of PERL. The #1 publisher of PERL reference books, of course is O'Reilly & Associates.

For the short run, especially as the OSS project is at the steepest part of it's growth curve, such investments generate positive ROI. Longer term, ROI motivations may steer these developers towards making proprietary extensions rather than releasing OSS.

Commoditizing Downstream Suppliers

This is very closely related to the loss leader business model However, instead of trying to get marginal service returns by massively growing the market, these businesses increase returns in their part of the value chain by commoditizing downstream suppliers.

The best examples of this currently are the thin server vendors such as Whistle Communications, and Cobalt Micro who are actively funding developers in SAMBA and Linux respectively.

Both Whistle and Cobalt generate their revenue on hardware volume. Consequently, funding OSS

enables them to avoid today's PC market where a "tax" must be paid to the OS vendor (NT Server retail price is $800 whereas Cobalt's target MSRP is around $1000).
The earliest Apache developers were employed by cash-strapped ISPs and ICPs.
Another, more recent example is IBM's deal with Apache. By declaring the HTTP server a commodity, IBM hopes to concentrate returns in the more technically arcane application services it bundles with it's Apache distribution (as well as hope to reach Apache's tremendous market share).
First Mover – Build Now, $$ Later
One of the exponential qualities of OSS – successful OSS projects swallow less successful ones in their space – implies a pre-emption business model where by investing directly in OSS today, they can pre-empt / eliminate competitive projects later – especially if the project requires API evangelization. This is tantamount to seizing a first mover advantage in OSS.
In addition, the developer scale, iteration rate, and reliability advantages of the OSS process are a blessing to small startups who typically can't afford a large in–house development staff.
Examples of startups in this space include SendMail.com (making a commercially supported version of the sendmail mail transfer agent) and C2Net (makes commercial and encrypted Apache)
Notice, that no case of a successful startup *originating* an OSS project has been observed. In both of these cases, the OSS project existed *before* the startup was formed.
Sun Microsystem's has recently announced that its "JINI" project will be provided via a form of OSS and may represent an application of the pre-emption doctrine.
Linux
The next several sections analyze the most prominent OSS projects including Linux, Apache, and now, Netscape's OSS browser.
A second memo titled "Linux OS Competitive Analysis" provides an in-depth review of the Linux OS. Here, I provide a top-level summary of my findings in Linux.
What is it?
Linux (pronounced "LYNN-ucks") is the #1 market share Open Source OS on the Internet. Linux is derives strongly from the 25+ years of lessons learned on the UNIX operating system.

Top-Level Features:

- Multi-user / Multi-threaded (kernel & user)
- Multi-platform (x86, Alpha, MIPS, PowerPC, SPARC, etc.)
- Protected 32-bit memory space for apps; Virtual Memory support (64-bit in development)
- SMP (Intel & Sun CPU's)
- Supports multiple file systems (FAT16, FAT32, NTFS, Ext2FS)
- High performance networking
  - NFS/SMB/IPX/Appletalk networking
  - Fastest stack in Unix vs Unix perf tests
- Disk Management
  - Striping, mirroring, FAT16, FAT32, NTFS
- Xfree86 GUI

Linux is a real, credible OS + Development process
Like other Open Source Software (OSS) products, the real key to Linux isn't the static version of the product but rather the process around it. This process lends credibility and an air of future-safeness to customer Linux investments.

- **Trusted in mission critical environments.** Linux has been deployed in mission critical, commercial environments with an excellent pool of public testimonials.
- **Linux = Best of Breed UNIX.** Linux outperforms many other UNIX's in most major performance category (networking, disk I/O, process ctx switch, etc.). To grow their featurebase, Linux has also liberally stolen features of other UNIX's (shell features, file systems, graphics, CPU ports)
- **Only Unix OS to gain market share.** Linux is on track to eventually own the x86

Open Source [redacted]ftware – 08/01/03; 4:19 PM

UNIX m[redacted]ket and has been the only UNIX version to gain net Server OS market share in recent years. I [redacted]lieve that Linux – moreso than NT – will be the biggest threat to SCO in the near future

Linux's process iterates VERY fast. For example, the Linux equivalent of the TransmitFile() API went from idea to final implementation in about 2 weeks time.

Linux is a short/medium-term threat in servers

The primary threat Microsoft faces from Linux is against NT Server.

Linux's future strength against NT server (and other UNIXes) is fed by several key factors:

- Linux uses commodity PC hardware and, due to OS modularity, can be run on smaller systems than NT. Linux is frequently used for services such as DNS running on old 486's in back closets.
- Due to it's UNIX heritage, Linux represents a lower switching cost for some organizations than NT
- UNIX's perceived Scalcability, Interopability, Availability, and Manageability (SIAM) advantages over NT.
- Linux can win as long as services / protocols are commodities

Linux is unlikely to be a threat on the desktop

Linux is unlikely to be a threat in the medium-long term on the desktop for several reasons:

- **Poor end-user apps & focus.** OSS development process are far better at solving individual component issues than they are at solving integrative scenarios such as end-to-end ease of use.
- **Switching costs for desktop installed base.** Switching desktops is hard and a challenger must be able to prove a significant marginal advantage. Linux's process is more focused on second-mover advantages (e.g. copying what's been proven to work) and is therefore unlikely to provide the first-mover advantage necessary to provide switching impetus.
- **UNIX heritage will slow encroachment.** Ease of use must be engineered from the ground up. Linux's hacker orientation will never provide the ease-of-use requirements of the average desktop user.

Beating Linux

In addition to the attacking the general weaknesses of OSS projects (e.g. Integrative / Architectural costs), some specific attacks on Linux are:

- Beat UNIX
  All the standard product issues for NT vs. Sun apply to Linux.
  Fold extended functionality into commodity protocols / services and create new protocols
  Linux's homebase is currently commodity network and server infrastructure. By folding extended functionality (e.g. Storage+ in file systems, DAV/POD for networking) into today's commodity services, we raise the bar & change the rules of the game.

Netscape

In an attempt to renew it's credibility in the browser space, Netscape has recently released and is attempting to create an OSS community around it's Mozilla source code.

Organization & LIcensing

Netscape's organization and licensing model is loosely based on the Linux community & GPL with a few differences. First, Mozilla and Netscape Communicator are 2 codebases with Netscape's engineers providing synchronization.

- Mozilla = the OSS, freely distributable browser
- Netscape Communicator = Branded, slightly modified (e.g. homepage default is set to home.netscape.com) version of Mozilla.

Unlike the full GPL, Netscape reserves the final right to reject / force modifications into the Mozilla codebase and Netscape's engineers are the appointed "Area Directors" of large components (for now).

Strengths

Capitalize on Anti-MSFT Sentiment in the OSS Community

Relative to other OSS projects, Mozilla is considered to be one of the most direct, near-term attacks on the Microsoft establishment. This factor alone is probably a key galvanizing factor in motivating developers

towards the Mozilla codebase.

New credibility

The availability of Mozilla source code has renewed Netscape's credibility in the browser space to a small degree. As BharatS points out in http://ie/specs/Mozilla/default.htm:

> "They have guaranteed by releasing their code that they will never disappear from the horizon entirely in the manner that Wordstar has disappeared. Mozilla browsers will survive well into the next 10 years even if the user base does shrink. "

Scratch a big itch

The browser is widely used / disseminated. Consequently, the pool of people who may be willing to solve "an immediate problem at hand" and/or fix a bug may be quite high.

Weaknesses

Post parity development

Mozilla is already at close to parity with IE4/5. Consequently, there no strong example to chase to help implicitly coordinate the development team.

Netscape has assigned some of their top developers towards the full time task of managing the Mozilla codebase and it will be interesting to see how this helps (if at all) the ability of Mozilla to push on new ground.

Small Noosphere

An interesting weakness is the size of the remaining "Noosphere" for the OSS browser.

1. The stand-alone browser is basically finished.
   There are no longer any large, high-profile segments of the stand-alone browser which must be developed. In otherwords, Netscape has already solved the interesting 80% of the problem. There is little / no ego gratification in debugging / fixing the remaining 20% of Netscape's code.
2. Netscape's commercial interests shrink the effect of Noosphere contributions.
   Linus Torvalds' management of the Linux codebase is arguably directed towards the goal of creating the best Linux. Netscape, by contrast, expressly reserves the right to make code management decisions on the basis of Netscape's *commercial / business interests*. Instead of creating an important product, the developer's code is being subjugated to Netscape's stock price.

Integration Cost

Potentially the single biggest detriment to the Mozilla effort is the level of integration that customers expect from features in a browser. As stated earlier, integration development / testing is NOT a parallelizable activity and therefore is hurt by the OSS process.

In particular, much of the new work for IE5+ is not just integrating components within the browser but continuing integration within the OS. This will be exceptionally painful to compete against.

Predictions

The contention therefore, is that unlike the Apache and Linux projects which, for now, are quite successful, Netscape's Mozilla effort will:

- Produce the dominant browser on Linux and some UNIX's
- Continue to slip behind IE in the long run

Keeping in mind that the source code was only released a short time ago (April '98), there is already evidence of waning interest in Mozilla. EXTREMELY unscientific evidence is found in the decline in mailing list volume on Mozilla mailing lists from April to June.

Mozilla Mailing List	April 1998	June 1998	% decline
Feature Wishlist	1073	450	58%
UI Development	285	76	73%
General Discussion	1862	687	63%

Internal mirrors of the Mozilla mailing lists can be found on http://egg.Microsoft.com/wilma/lists

Open Source [redacted]ftware – 08/01/03; 4:19 PM

Apache
History
Paraphrased f[redacted]m http://www.apache.org/ABOUT_APACHE.html

> In February of 1995, the most popular server software on the Web was the public domain HTTP daemon developed by NCSA, University of Illinois, Urbana-Champaign. However, development of that httpd had stalled after mid-1994, and many webmasters had developed their own extensions and bug fixes that were in need of a common distribution A small group of these webmasters, contacted via private e-mail, gathered together for the purpose of coordinating their changes (in the form of "patches"). By the end of February '95, eight core contributors formed the foundation of the original Apache Group. In April 1995, Apache 0.6.2 was released.
>
> During May-June 1995, a new server architecture (code-named Shambhala) was developed which included a modular structure and API for better extensibility, pool-based memory allocation, and an adaptive pre-forking process model. The group switched to this new server base in July and added the features from 0.7.x, resulting in Apache 0.8.8 (and its brethren) in August.
>
> Less than a year after the group was formed, the Apache server passed NCSA's httpd as the #1 server on the Internet.

Organization
The Apache development team consists of about 19 core members plus hundreds of web site administrators around the world who've submitted a bug report / patch of one form or another. Apache's bug data can be found at: http://bugs.apache.org/index.
A description of the code management and dispute resolution procedures followed by the Apache team are found on http://www.apache.org:
Leadership:

> There is a core group of contributors (informally called the "core") which was formed from the project founders and is augmented from time to time when core members nominate outstanding contributors and the rest of the core members agree.

Dispute resolution.

> Changes to the code are proposed on the mailing list and usually voted on by active members -- three +1 (yes votes) and no -1 (no votes, or vetoes) are needed to commit a code change during a release cycle

Strengths
Market Share!
Apache far and away has #1 web site share on the Internet today6. Possession of the lion's share of the market provides extremely powerful control over the market's evolution.
In particular, Apache's market share in web server space presents the following competitive hurdles:

- Lowest common denominator HTTP protocol – slows our ability to extend the protocol to support new applications
- Breathe more life into UNIX – Where Apache goes, Unix must follow.

3rd Party Support
The number of tools / modules / plug-ins available for Apache has been growing at an increasing rate.
Weaknesses
Performance
In the short run, IIS soundly beats Apache on SPECweb. Moving further, as IIS moves into kernel and

---

6 As anyone in the IIS team will quickly note, however, there is a large difference between "site" and "server" since multiple sites may be run on a single server. Gspix points out that Apache only serves ~2% of all web page "hits"

takes advantage deeper integration with the NT, this lead is expected to increase further.
Apache, by contrast, is saddled with the requirement to create portable code for all of its OS environments.

HTTP Protocol Complexity & Application services

Part of the reason that Apache was able to get a foothold and take off was because the HTTP protocol is so simple. As more and more features become layered on top of the humble web server (e.g. multi-server transaction support, POD, etc.) it will be interesting to see how the Apache team will be able to keep up. ASP support, for example is a key driver for IIS in corporate intranets

IBM & Apache

Recently, IBM announced it's support for the Apache codebase in its WebSphere application server. The actual result of the press furor is still unclear however:

- IBM still ships and supports both Apache and Domino's GO web server
- IBM's commitment appears to be:
  - Helping Apache port to strategic IBM platforms (AS/400, etc.)
  - Redistributing Apache binaries to customers who request Apache support
  - Support for Apache binaries (only if they were purchased through IBM?)
- IBM has developers actively participating in Apache development / discussion groups.
- IBM is taking a lead role in optimizing Apache for NT

Other OSS Projects

Some other OSS projects:

- **Gimp** – http://www.gimp.org – Gimp (GNU Image Manipulation Program) is an OSS project to create an Adobe Photoshop clone for Unix workstations. Feature-wise, however, their version 1.0 project is more akin to PaintBrush.
- **WINE / WABI** – http://www.wine.org – Wine (Wine Is Not an Emulator) is an OSS windows emulation library for UNIX. Wine competes (somewhat) with Sun's WABI project which is non-OSS. Older versions of Office, for example, are able to run in WINE although performance remains to be evaluated.
- **PERL** – http://www.perl.org – PERL (Practical Evaluation and Reporting Language) is the defacto standard scripting language for all Apache web servers. PERL is very popular on UNIX in particular due to its powerful text/string manipulation and UNIX's reliance on command line administration of all functionality.
- **BIND** – http://www.bind.org – BIND (Berkeley Internet Name Daemon) is the de facto DNS server for the Internet. In many respects, DNS was developed on top of BIND.
- **Sendmail** – http://www.sendmail.org – Sendmail is the #1 share mail transfer agent on the Internet today.
- **Squid** – http://www.squid.org – Squid is an OSS Proxy server based on the ICP protocol. Squid is somewhat popular with large international ISPs although it's performance is lacking.
- **SAMBA** – http://www.samba.org – SAMBA provides an SMB file server for UNIX Recently, the SAMBA team has managed to reverse engineer and develop an NT domain controller for UNIX as well. SGI employs one of the SAMBA leads. http://www.sonic.net/~roelofs/reports/linux-19980714-phq.html: *"By the end of the year ... Samba will be able to completely replace all primary NT Server functions."*
- **KDE** – http://www.kde.org – "K" Desktop Environment. Combines integrated browser, shell, and office suite for Unix desktops. Check out the screen shots at: http://www.kde.org/kscreenshots.html and http://www.kde.org/koffice/index.html.
- **Majordomo** – the dominant mail list server on the Internet is writtenentirely in PERL via OSS.

Microsoft Response

In general, a lot more thought/discussion needs to put into Microsoft's response to the OSS phenomena. The goal of this document is education and analysis of the OSS process, consequently in this section, I present only a very superficial list of options and concerns.

Open Source [redacted]ftware – 08/01/03; 4:19 PM

Product Vuln[redacted]abilities
Where is Mic[redacted]soft most likely to feel the "pinch" of OSS projects in the near future?
Server vs. Cli[redacted]t
The server is more vulnerable to OSS products than the client. Reasons for this include:

- **Clients "task switch" more often** – the average client desktop is used for a wider variety of apps than the server. Consequently, integration, ease-of-use, fit & finish, etc. are key attributes
- **Servers are more task specific** – OSS products work best if goals/precedents are clearly defined – e.g. serving up commodity protocols
- **Commodity servers are a lower "commitment" than clients** – Replacing commodity servers such as file, print, mail-relay, etc. with open source alternatives doesn't interfere with the end-user's experience. Also, in these commodity services, a "throw-away" "experimental" solution will often by entertained by an organization.
- **Servers are professionally managed** – This plays into OSS's strengths in customization and mitigates weaknesses in lack of end-user ease of use focus.7

Capturing OSS benefits -- Developer Mindshare
The ability of the OSS process to collect and harness the collective IQ of thousands of individuals across the Internet is simply amazing. More importantly, OSS evangelization scales with the size of the Internet much faster than our own evangelization efforts appear to scale.
How can Microsoft capture some of the rabid developer mindshare being focused on OSS products?
Some initial ideas include:

- **Capture parallel debugging benefits via broader code licensing** – Be more liberal in handing out source code licenses to NT to organizations such as universities and certain partners.
- **Provide entry level tools for low cost / free** – The second order effect of tools is to generate a common skillset / vocabulary tacitly leveraged by developers As NatBro points out, the wide availability of a consistent developer toolset in Linux/UNIX is a critical means of implicitly coordinating the system.
- **Put out parts of the source code** – try to generate hacker interest in adding value to MS-sponsored code bases. Parts of the TCP/IP stack could be a first candidate. OshM points out, however that the challenge is to find some part of MS's codebase with a big enough Noosphere to generate interest.
- **Provide more extensibility** – The Linux "enthusiast developer" loves writing to / understanding undocumented API's and internals. Documenting / publishing some internal API's as "unsupported" may be a means of generating external innovations that leverage our systems investments. In particular, ensuring that more components from more teams are scriptable / automatable will help ensure that power users can play with our components.
- **Creating Community/Noosphere**. MSDN reaches an extremely large population. How can we create social structures that provide network benefits leveraging this huge developer base? For example, what if we had a central VB showcase on Microsoft.com which allowed VB developers to post & published full source of their VB projects to share with other VB developers? I'll contend that many VB developers would get extreme ego gratification out of having their name / code downloadable from Microsoft.com.
- **Monitor OSS news groups**. Learn new ideas and hire the best/brightest individuals.

Capturing OSS benefits – Microsoft Internal Processes
What can Microsoft learn from the OSS example? More specifically: How can we recreate the OSS development environment internally? Different reviewers of this paper have consistently pointed out that internally, we should view Microsoft as an idealized OSS community but, for various reasons do not:

---

7 George Spix points out that in the long run, the structure of the IT department will become more like the end-user computing: "There are only 14,000 profit making companies in the US with over 500 employees and the number is declining Commodity IT is here. What applies to the client applies to the server."

- **Different development "modes".** Setting up an NT build/development environment is extremely complex & wildly different from the environment used by the Office team.
- **Different tools / source code managers.** Some teams use SLM, other use VSS Different bug databases. Different build processes.
- **No central repository/code access.** There is no central set of servers to find, install, review the code from projects outside your immediate scope. Even simply providing a central repository for debug symbols would be a huge improvement. NatBro: "a developer at Microsoft working on the OS can't scratch an itch they've got with Excel, neither can the Excel developer scratch their itch with the OS – it would take them months to figure out how to build & debug & install, and they probably couldn't get proper source access anyway"
- **Wide developer communication.** Mailing lists dealing with particular components & bug reports are usually closed to team members.
- **More component robustness.** Linux and other OSS projects make it easy for developers to experiment with small components in the system without introducing regressions in other components: DavidDs: "People have to work on their parts independent of the rest so internal abstractions between components are well documented and well exposed/exported as well as being more robust because they have no idea how they are going to be called. The linux development system has evolved into allowing more devs to party on it without causing huge numbers of integration issues because robustness is present at every level. This is great, long term, for overall stability and it shows."

The trick of course, is to capture these benefits without incurring the costs of the OSS process These costs are typically the reasons such barriers were erected in the first place:

- **Integration.** A full-time developer on a component has a lot of work to do already before trying to analyze & integrate fixes from other developers within the company.
- **Iterative costs & dependencies.** The potential for mini-code forks between "scratched' versions of the OS being used by one Excel developer and "core" OS used by a different Excel developer.

Extending OSS benefits -- Service Infrastructure

Supporting a platform & development community requires a lot of *service* infrastructure which OSS can't provide. This includes PDC's, MSDN, ADCU, ISVs, IHVs, etc.

The OSS communities "MSDN" equivalent, of course, is a loose confederation of web sites with API docs of varying quality. MS has an opportunity to <u>really</u> exploit the web for developer evangelization.

Blunting OSS attacks

Generally, Microsoft wins by attacking the core weaknesses of OSS projects.

De-commoditize protocols & applications

OSS projects have been able to gain a foothold in many server applications because of the wide utility of highly commoditized, simple protocols. By extending these protocols and developing new protocols, we can deny OSS projects entry into the market.

David Stutz makes a <u>very</u> good point: in competing with Microsoft's level of desktop integration, "*commodity protocols actually <u>become</u> the means of integration*" for OSS projects. There is a large amount of IQ being expended in various IETF working groups which are quickly creating the architectural model for integration for these OSS projects.

Some examples of Microsoft initiatives which are extending commodity protocols include:

- **DNS integration with Directory**. Leveraging the Directory Service to add value to DNS via dynamic updates, security, authentication
- **HTTP-DAV**. DAV is complex and the protocol spec provides an infinite level of implementation complexity for various applications (e.g. the design for Exchange over DAV is good but certainly not <u>the</u> single obvious design). Apache will be hard pressed to pick and choose the correct first areas of DAV to implement.
- **Structured storage**. Changes the rules of the game in the file serving space (a key

Linux/Ap[redacted]he application). Creates a compelling client-side advantage which can be extended to the server as [redacted]ell

- **MSMQ for Distributed Applications**. MSMQ is a great example of a distributed technology where most of the value is in the services and implementation and NOT in the wire protocol. The same is true for MTS, DTC, and COM+.

Make Integration Compelling – Especially on the server

The rise of specialty servers is a particularly potent and dire long term threat that directly affects our revenue streams. One of the keys to combating this threat is to create integrative scenarios that are valuable on the server platform. David Stutz points out:

> The bottom line here is whoever has the best network-oriented integration technologies and processes will win the *commodity server business*. There is a convergence of embedded systems, mobile connectivity, and pervasive networking protocols that will make the number of servers (especially "specialist servers"??) explode. The general-purpose commodity client is a good business to be in - will it be dwarfed by the special-purpose commodity server business?

- **System Management**. Systems management functionality potentially touches all aspects of a product / platform. Consequently, it is not something which is easily grafted onto an existing codebase in a componentized manner. It must be designed from the start or be the result of a conscious re-evaluation of all components in a given project.
- **Ease of Use**. Like management, this often must be designed from the ground up and consequently incurs large development management cost. OSS projects will consistently have problems matching this feature area
- **Solve Scenarios**. ZAW, dial up networking, wizards, etc.
- **Client Integration**. How can we leverage the client base to provide similar integration requirements on our servers? For example, MSMQ, as a piece of middleware, requires closely synchronized client and server codebases.
  Middleware control is critical. Obviously, as servers and their protocols risk commoditization higher order functionality is necessary to preserve margins in the server OS business.

Organizational Credibility

- **Release / Service pack process.** By consolidating and managing the arduous task of keeping up with the latest fixes, Microsoft provides a key customer advantage over basic OSS processes.
- **Long-Term Commitments.** Via tools such as enterprise agreements, long term research, executive keynotes, etc., Microsoft is able to commit to a long term vision and create a greater sense of long term order than an OSS process.

Other Interesting Links

- http://www.lwn.net/ -- summarizes the weeks events in Linux development world.
- Slashdot – http://slashdot.org/ – daily news / discussion in the OSS community
- http://www.linux.org
- http://www.opensource.org
- http://news.freshmeat.net/ -- info on the latest open source releases & project updates

Acknowledgments

Many people provided, datapoints, proofreading, thoughtful email, and analysis on both this paper and the Linux analysis:

Nat Brown	George Spix	Dwight Krossa
Jim Allchin	David Stutz	David D'Souza
Charlie Kindel	Stephanie Ferguson	David Treadwell
Ben Slivka	Jackie Erickson	David Gunter
Josh Cohen	Michael Nelson	Oshoma Momoh

[redacted]pen Source Software --08/01/03; 4:19 PM

[redacted]lex Hopman Sankar Koundinya Bernard Aboba
[redacted]ffrey Robertson Alex Sutton

Revision History

Date	Revision	Comments
8/03/98	0.95	
8/10/98	0.97	Started revision table Folded in comments from JoshCo
8/11/98	1.00	More fixes, printed copies for PaulMa review

Microsoft Corporation

Linux OS Competitive Analysis

The Next Java VM?

Vinod Valloppillil (VinodV)

Josh Cohen (JoshCo)

Aug 11, 1998 – v1.00

Microsoft Confidential

Table of Contents

Linux Operating System

The Next Java VM?

Executive Summary

The Linux OS is the highest visibility product of the Open Source Software (OSS) process. Linux represents a best-of-breed UNIX, that is trusted in mission critical applications, and – due to it's open source code – has a long term credibility which exceeds many other competitive OS's.

Linux poses a significant near-term revenue threat to Windows NT Server in the commodity file, print and network services businesses. Linux's emphasis on serving the hacker and UNIX community alleviates the near-medium term potential for damage to the Windows client desktop.

In the worst case, Linux provides a mechanism for server OEMs to provide integrated, task-specific products and completely bypassing Microsoft revenues in this space.

[This document assumes that the reader has read the "Open Source Software" doc first. Many of the ideas / assertions here are derived from the previous doc and many other applicable Open Source arguments are not repeated here for brevity.]

Linux History

What is it?

Linux (pronounced "LYNN-ucks") is the #1 market share Open Source OS on the Internet. Linux derives strongly from the 25+ years of lessons from the UNIX operating system.

Top-Level Features:

- Multi-user / Multi-threaded (kernel & user)
- Multi-platform (x86, Alpha, MIPS, PowerPC, SPARC, etc.), source compatibility
- Protected 32-bit memory space for apps; Virtual Memory support;
- 64-bit support (platform dependent)
- SMP (Intel & Sun CPU's)
- Supports multiple file systems (FAT16, FAT32, NTFS, various UNIX)
- High performance networking
  - NFS/SMB/IPX/Appletalk networking
  - Fastest stack in Unix vs. Unix performance tests
- Disk Management
  - Striping, mirroring, RAID 0,1,5
- Xfree86 GUI

History

An excellent piece on the history of the Linux Operation system is provided by Wired Magazine at http://www.wired.com/wired/5.08/linux.html. I've paraphrased some of the key points below.

Linux was original the brainchild of Linus Torvalds, an undergraduate student at the University of Helsinki. In addition to a 80386-based kernel, Linus wrote keyboard and screen drivers to attach to PC hardware and provided this code under GNU's Public License on an FTP site in the summer of 1991.

After hosting his work on the FTP site, he announced it's availability on a Minix USENET discussion group in late summer 1991. By January of 1992, over 100 users / hackers had downloaded Linux and -- more importantly -- were regularly contributing / updating the source code with new fixes, device drivers, etc.

In contrast to the FSF/GNU work, which provided developers an open source abstraction above the underlying, commercial UNIX OS kernel, Linux's team was creating a completely open source kernel. In time, more and more of the GNU user/shell work was ported to Linux to round out the platform for hackers.

Forbes magazine's story on Linux has some excellent data on Linux's development history http://www.forbes.com/forbes/98/0810/6209094s1.htm:

Date	Users	Version	Size (LOC)
1991	1	0.01	10k
1992	1000	0.96	40k
1993	20,000	0.99	100k
1994	100,000	1.0	170k
1995	500,000	1.2	250k
1996	1.5M	2.0	400k
1997	3.5M	2.1	800k
1998	7.5M	2.1.110	1.5M

The LOC count appears to be inclusive of all Linux ports including x86, PPC, SPARC, etc.

Linux 1.0 - March 1994

Linux 1.0 was the first major release and led to the creation of "distributions." Prior to 1.0, linux existed as a piecemeal kernel with no centralized place to get a full working OS.

Major Features:

- Virtual Memory Management / memory Mapping / Buffer cache
- Job Control
- Device support for popular Network Cards, Hard Drives, CDROMs, etc
- Named Pipes, IPC
- Original EXTFS support instead of Minixfs
- Preemptive multitasking

Management Structure

After the release of version 1.0. The Linux developer community adopted a management structure to control what is added to the kernel with even numbered releases as stable, production release branches and odd numbered versions were "developer" branches.

While major areas of the kernel have "owners" which maintain their areas, Linus remains the final say on what does and does not go into the kernel. In large part, this structure remains in place.

It is important to distinguish that this management structure only controls the actual kernel and does not include supporting areas like the GUI, system utilities and servers, and system libraries.

Since 1.0, the following 1.x branches existed:
1.1 3/95
1.2 8/95
1.3 6/96

Version 1.3 evolved to become version 2.0

Linux 2.0 – June 1996

Linux v2.0 was the first major release could effectively compete as a UNIX distribution. The kernel, system libraries, the GNU Unix tool, X11, various open source server applications such as BIND and sendmail, etc. were frozen and declared part of Linux 2.0.

Around the same time the GNU/FSF agreed, reluctantly, to make the Linux kernel the official kernel of the GNU operating system[1].

Some of the new base libraries and tools:

- Kernel modules 2.0.0 – Basic kernel module support
- PPP daemon 2.2.0f - Dialup networking
- Dynamic linker (ld.so) 1.7.14 – Shared libraries
- GNU CC 2.7.2 – C compiler, tools, and debugger
- Binutils 2.6.0.14 – Support for various binary executable formats
- Linux C Library Stable: 5.2.18,
- Linux C++ Library 2.7.1.4
- Termcap 2.0.8 – Console mode terminal drivers
- Procps 1.01 – ProcFS file system maps kernel objects to the filesystem
- SysVinit 2.64 – A system V boot system, SYSV compliant named pipes.
- Net-tools 1.32-alpha- Basic Networking tools such as telnet, finger, etc
- Kbd 0.91 – Console mode keyboard/scrollback/ virtual screens support

Subsequent Versions

The current 2.0.x stable version is 2.0.34, which was released in May 1998. Prior to this, 2.0.33 was released in Dec 1997. The current development branch is 2.1.108 (as of July 14, 1998).

[1] Prior to this, Stallman and Co. were developing the HURD kernel which languished in limbo

Process Slowdown

With the growth of the kernel, Linux's release frequency has slowed measurably. There is growing frustration about when 2.2, the next "stable release" version will ship. The sheer size of the codebase has begun to overrun the resources of Linus. There is a backlog of patches to be merged and often, Linus is becoming the choke point.

The current release tree, 2.0.x has iterated 34 versions in 2 years. The development branch, 2.1.x, which will eventually become 2.2 has been going on since 9/96 spanning 108 versions and no ship date in sight.

Even though the feature freeze is declared, major changes continue to get integrated into the kernel. Most merges seem to be due to fundamental bug fixes and or cross platform issues.

Organization

An analysis / description of the OSS development organization and process is in a second memo titled "Open Source Software." This section describes attributes of OSS that are unique to Linux.

Wired Magazine ran a recent story chronicling the history of Linux "The Greatest OS that (N)ever was" http://www.wired.com/wired/5.08/linux.html.

> *The growth of the development team mirrored the organic, not to say chaotic, development of Linux itself. Linus began choosing and relying on what early Linux hacker Michael K. Johnson calls "a few trusted lieutenants, from whom he will take larger patches and trust those patches. The lieuts more or less own relatively large pieces of the kernel."*

As with other OSS projects, the General Public License ("CopyLeft") and it's relatives are considered instrumental towards creating the dynamic behavior around the Linux codebase:

> *In a sense, GPL provided a written constitution for the new online tribe of Linux hackers. The license said it was OK to build on, or incorporate wholesale, other people's code - just as Linux did - and even to make money doing so (hackers have to eat, after all). But you couldn't transgress the hacker's fundamental law of software: source code must be freely available for further hacking*

Linux Technical Analysis & OS Structure

Anatomy of a Distribution

"Linux" is technically just a kernel, not the entire supporting OS. In order to create a usable product, Linux "distributions" are created which bundle the kernel, drivers, apps and many other components necessary for the full UNIX/GUI experience.

These subsystems are typically developed in an OSS manner as well and several of them – e.g. the Xfree86 GUI – have a codebase size/complexity that exceeds the Linux kernel.

These external components come from many sources and are individually hand picked by the distribution vendor for a particular product. A frequent source of controversy stems from distribution vendors bundling non-GPL code with the Linux kernel and mass distributing them.

A partial list of components is in the following table:

Component	Codebase / Name	Provider/Maintainer(s)
Kernel	Basic OS, Networking Stack	Linux (http://www.kernel.org)
File System(s)	Msdos, ext2fs	Linux Kernel
Sys Libs	Glibc, Lib5c	GNU / FSF
Drivers		Linux, Individual Contributors
User Tools	Gnu user tools	GNU/FSF
System Installation	LISA	Caldera
App Install Management	RedHat Package Manager	RedHat
Development Tools	GNU Development tools GCC	GNU/FSF
Web Server	APACHE	The Apache Group http://www.apache.org/
Mail Server	SendMail	http://www.sendmail.org
DNS Server	BIND	http://www.bind.org
SMB Server	SAMBA	http://www.samba.org
X Server	Xffree86 / MetroX	Xfree86 project / MetroX commercial
Window Manager	FVWM	GPL
Widgets	Motif	X Consortium
Desktop Tools	X Contrib KDE Gnome	X Consortium http://www.kde.org http://www.gnome.org
Management	RPM Package Installed Roll own distribution specific	RedHat (free) Debian / Slackware

Descriptions of some of the larger components are below:

Kernel – GPL

The kernel is the core part of Linux that is expressly managed by Linus and his lieutenants and is protected via the GPL.

Functions contained in the Linux Kernel include:

- Core OS Features (scheduling, memory management, threads, Hardware Abstraction, etc)
- Network Stack
- File system

Extensive on-line documentation of the Linux kernel architecture and components can be found on: http://sunsite.unc.edu/linux/LDP/tlk/tlk.html. Note that video drivers exist outside the kernel – the kernel only has rudimentary text display support to a console.

Drivers – GPL

An assortment of modules for standard functions and devices are typically part of the kernel distribution. In addition, a selection of non-standard modules is often included. Mostly GPL, however in some cases, NDAs with hardware manufacturers are required to get specs to make a driver, in which case they are not open source.

Linux device drivers are typically developed by users for specific devices on their machines. This incremental, piecemeal process has created a very large pool of device drivers for Linux (as of 7/1/98):

- Video: http://sunsite.unc.edu/LDP/HOWTO/Hardware-HOWTO-6.html – close to 400 drivers available
- Network: http://sunsite.unc.edu/LDP/HOWTO/Hardware-HOWTO-11.html – ~75 network cards supported
- PCMCIA http://sunsite.unc.edu/LDP/HOWTO/Hardware-HOWTO-26.html – ~150 supported cards.

NatBro points out:

> An important attribute to note which has led to volume drivers is the ease with which you can write drivers for linux, and the relatively powerful debugging infrastructure that linux has. Finding and installing the DDK, and trying to hook up the kernel debugger and do any sort of interaction with user-mode without tearing the NT system to bits is much more challenging than writing the simple device-drivers for linux. Any idiot could write a driver in 2 days with a book like "Linux Device Drivers" – there is no such thing as a 2-day device-driver for NT

Recently, a small number of hardware vendors have begun to provide Linux drivers for their NICs (3Com) and SCSI adapters (Adaptec). These drivers are believed to be protected by the Library-GPL and are consequently not open source (the Library-GPL is described later). It remains to be seen whether this will create the momentum to develop more commercial drivers for Linux.

System Libraries & Apps – GNU GPL

System libraries provide:

- Basic POSIX api's for system services
- Basic API's to support commandline / shell utilities.

The system libraries in a Linux distribution are NOT managed by Linus. As such, there has been a small amount of versioning / forking in this area with two dominant libraries – glibc and lib5c which introduce minor incompatibilities between different apps.

User Tools (GPL, GNU FSF)

These are basic UNIX command line tools and shell environments. Many shell environments exist although all are supported by the FSF.

Also included in this category are "old standby" apps such as finger, telnet, etc.

Development Tools (GPL)

A hallmark of the UNIX operating system is the free availability of development tools / compilers. The GCC and PERL language compilers are often provided for free with all versions of Linux and are available for other UNIXes as well.

These tools are the "old standbys" of the UNIX development world and are widely used across all Unix platforms. This mass commoditization of development/debug tools is a key contributor to the common skillset efficiencies realized by the Linux process.

By the standards of the novice / intermediate developer accustomed to VB/VS/VC/VJ, these tools are incredibly primitive.

GUI / UI

X Server

The X Server standard is owned by MIT under contract by the X Consortium. X Consortium's licensing practices are viewed as too restrictive by the OSS crowd so a series of public X initiatives were launched with Xfree86 being the dominant distribution.

Interestingly, the Xfree86 development team licenses their code under the BSD license because they consider GPL too restrictive: http://www.redhat.com/linux-info/xfree86/developer.html.

Configuring the XFree86 system on Linux can be a very difficult, time consuming process. Linux has no hardware abstraction layer for video services, and most video card manufacturers do not provide Linux OS video drivers. Thus, XFree86 provides internal support for a wide variety of video cards and chipsets. Correctly configuring XFree86 requires the user to know the manufacturer, model, and chipset for their video card. In many cases, the user must know or calculate the video timings as well.

Widgets & Desktops

There are multiple widget sets which exist in many applications, so all X applications do not look the same or act the same ways like in Windows. Motif is considered the defacto Unix widget set, but since it is not freely distributable, it is contrary to the Linux model.

Consequently, Linux distributions usually choose one of several similar, but not completely compatible Widget sets.

- Motif
- LessTif
- Xaw3d (3d athena widgets that look like motif)
- QT

Obviously, this mess has spawned several efforts to unify the "desktop" as well as the widget sets. In typical Linux fashion, there are several competing efforts:

- Gnome/totally new
- KDE
- FreeQT/KDE
- CDE/commercial

Commercial Linux OS

Binary Compatibility

Server

Almost all of the system components necessary to run server applications are part of the core distribution maintained by Linus. Consequently, for a given hardware type, almost all Linux server application binaries will natively run. Across hardware types (e.g. x86 vs. PPC), generally only a recompile of the application is necessary.

There is essentially 100% source code compatibility for system application code.

Solaris / SCO x86 Compatibility

Via compatibility libraries, Linux on x86 is able to natively execute most SCO UNIX and Solaris x86 binaries. Oracle on SCO is widely cited as an example (although Oracle does not "officially" support SCO binaries on top of Linux – also Oracle has recently announced development of a native Linux version of Oracle 8 to ship in March 1999)

Client

Client distributions, however, are a different story stemming most directly from the current "mess" in X-windows / GUI systems for Linux.

Binary compatibility issues generally stem from differences in non-kernel code that's required to turn the kernel into a full OS.

Binary Incompatibility: Netscape Communicator

One example of this incompatibility is Netscape Communicator for Linux. The released versions of Netscape Communicator for Linux are built based on libc5, instead of the newer glibc which Caldera supports. RedHat, however ships glibc instead of libc5 requiring users install libc5 as well as glibc.

RedHat

http://www.redhat.com

RedHat Corporation was founded in 1995 by a pair of Linux developers/enthusiasts with the intent of creating a commercially supported, "cleaned-up" Linux distribution.

The company currently has ~35 employees. Financials and some run-rate information is available in an interview with their CEO in Infoworld (http://www.infoworld.com/cgi-bin/displayArchive.pl?/98/23/e03-23.102.htm):

> Bob Young, president of Red Hat expects the 3-year old company to earn revenues of $10 million this year and to ship about 400,000 copies of Linux, ranging from $50 to near $1,000 for a supported version.

Commercially-Developed Extensions

Perhaps the most interesting aspect of Red Hat's business model is their extremely active and continuing contributions to the Linux community. Several prior initiatives spearheaded by RedHat have been released as OSS for modification. In most cases, these code releases were simple fixes or additional drivers.

Redhat actively employs several key Linux developers and pays them to hack Linux fulltime. Some of the components which have been "donated" back to the Linux effort include:

- **RedHat Package Manager** -- RPM is Linux component which provides application install / maintenance facilities for Linux similar to the Application Manifest being developed by Microsoft.
- **Pluggable-Authentication Manager** -- PAM is similar to the NT SSPI / SAM system and allows for componentized plug-ins to handle the authentication function (RedHat provides an LDAP plugin). PAM was originally available on Sun systems.

One of the larger "grants" however has been the now universal "Redhat Package Manager" or RPM which ships with almost all Linux distributions. RPM creates the concept of an application manifest which simplifies the job of installing & removing applications on top of Linux.

Redhat's current development project is a new GUI for Linux call "Gnome". Gnome is a response to latent concerns with non-GPL versions of the X-windows user interface.

Product Features

Of the Commercial Linux Distributors, Redhat has the largest array of SKU's. At the highest end, Redhat bundles the following with their distributions of Linux:

- Apache Web Server
- Corel WordPerfect
- DBMaker DBMS by Casemaker
- Xfree86 window server

Caldera

Caldera is Ray Noorda's latest company with its eye on the operating system marketplace. Caldera's financials and sales are unpublished but it is widely believed to be the #2 commercial Linux vendor after RedHat.

Caldera bundles several components with their version of Linux including:

- StarOffice 4.0 by Germany's Star Corp.

- Adabas SQL Server by Software AG
- Netware client & Admin
- Netscape fasttrack server + communicator
- Xfree86 and MetroX X-window systems

Others

Other Linux distributions seem to be falling by the wayside of RedHat and Caldera. They include SlackWare, SuSe, and Debian to name a few. A comprehensive list of distributions can be found on http://www.linux.org.

Commercial Linux ISV's

There are currently no major ISV's who derive a significant percentage of their sales from the Linux platform. A somewhat complete list of the commercial apps available on Linux can be found on: http://www.uk.linux.org/LxCommercial.html.

Reasons for this include:

- **First-use Linux apps are free** – most of the primary apps that people require when they move to Linux are already available for free. This includes web servers, POP clients, mail servers, text editors, etc.

- **Linux market is still immature** – the Linux market is still in its infancy and the current state of Linux commercial software may change radically in the coming months

- **Current Linux users are wary of commercial products** – you can scout any of various Linux discussion and mailing lists and quickly run into users admonishing commercial software providers and trying to launch a jihad against category X via open source software (at the time of this writing, Lotus Notes is a popular target)[2]

Library-GPL

Unlike the GPL (General Public License – described in depth in "Open Source Software") which forces all derivative works to be free, Linux software libraries have the more limited "Library GPL" which allows applications which merely link to Linux to be considered non-derivative.

The Library-GPL removes a key impediment to commercial software vendors developing products on top of Linux.

The Library-GPL is defined at http://www.fsf.org/copyleft/lgpl.html

Binary Unix Compatibility

Linux adheres to several UNIX standards most notably POSIX 1003.1c. When compiled and running on it's various CPU platforms, Linux is generally binary compatible (more so on the server than on the desktop) with the primary commercial UNIXs including:

[2] An IBM official in describing IBM's hesitation to support Linux as an app platform stated: "Linux does not run in many corporations that actually buy product." http://www.techweb.com/se/directlink.cgi?CRN19980727S0127

- Solaris/SunOS on SPARC
- Solaris on x86
- SCO on x86
- Digital UNIX on Alpha
- SGI IRIX on MIPs

Microsoft

Microsoft's current involvement in Linux is limited to distribution of client code for strategic services such as Netshow as well as helping SAG port DCOM to Linux. IE is currently not officially supported on Linux.

Intel

Intel is directly involved in helping port Linux to Merced. Intel is also involved with the GCC over Merced development efforts.

Netscape

In the press, Netscape is sited as the #1 commercial provider of software for Linux. Marc Andreeson has been extensively quoted as saying that "Linux is a tier 1 platform for Netscape".

Until recently, however, the only server product that Netscape explicitly sells for Linux is their Fasttrack server with other servers merely being licensed to the respective Linux vendors for their own redistribution. On July 21st, however, Netscape formally announced intentions to port all of their server application products to Linux starting with Mail and Directory services.

All of Netscape's client products are available on the Linux platform.

Oracle

Oracle recently announced (7/18/98) their support for Oracle 8 on top of Linux to be shipped in March 1999.

Sun

Sun's involvement in Linux is inconclusive. Early this year (1998), Sun joined the board of Linux International which is one of many user groups representing Linux.

At one level, Linux competes (quite favorably) against Sun's own Solaris x86 port.

At a secondary level, Sun may view Linux as a strategic ally b/c it generally represents the low-end of the software market and could therefore arguably hurt Microsoft more than it hurts Sun.

SoftwareAG

SoftwareAG has ported it's ADABAS database server to Linux and is currently bundled with Caldera's distribution.

Corel

Corel has ported their WordPerfect Suit to Linux and is currently offering it bundled with several of RedHat's SKU's

Computer Associates

Recently announced intentions to port CA-Ingres DB to Linux: http://x10.dejanews.com/getdoc.xp?AN=370037691&CONTEXT=900053229.949289093&hitnum=0.

Market Share

Linux's exact market share is very difficult to calculate because:

- The majority of Linux installations are downloaded from anonymous FTP sites – NOT purchased. Consequently, there are no published sales figures to track.
- (Some) Commercial Linux purchases can be used to install multiple machines
- Because Linux revs so often, there's a very high likelihood of double-counting actual installations vs. downloads/purchases
- There are no separate client & server distributions. Consequently, it's difficult to compare Linux numbers wholesale to NTS / NTW numbers without accurate usage data from the Linux community.

Below I include data / pointers from some of the more prominent attempts to isolate the number of Linux users.

Installed Base

The most comprehensive Linux market share survey was published by Red Hat in March 1998: http://www.redhat.com/redhat/linuxmarket.html

Using available data collected from other distributions, RedHat calculated a retail CD sell rate of :

- 1996: **450,000**
- 1997: **750,000**

RedHat's estimate of the growth of the Linux installed user base (which includes CD purchases as well as downloads as well as client + server) is:

- 1993: **100k**
- 1994: **500k**
- 1995: **1.5M**
- 1996: **3.5M**
- 1997: **7.5M**

Other estimates put the Linux installed base from 5 Million (Ziff Davis), to 10 Million (Linux advocates).

Server

IDC's most recent "Server Operating Environments" report provides the following breakdown of shipments in the Server OS space.

		1996	1997	growth
NTS	units	805,200	1,505,000	86.91%
	%	23.30	36.6	
Netware	units	993,000	900,000	-9 37%
	%	28.7	21.9	
Linux	units			
	%	5.8	5.8	
Solaris	units	81,000	99,500	22.84%
(combined x86/SPARC)	%	2.34389	2.4	
SCO	units	226,000	288000	27.43%
(combined OpenServer/Unixware)	%	6.5	7	
Other	units	1,150,537	1,079,478	-6.18%
	%	39.3	33.3	
Totals	units	3,455,794	4,112,022	18.99%
	%	100	100	

Using the 240K number shipped in 1997, IDC seems to be estimating ~750K total installed Linux *server* systems. Compared to other market share studies, IDC's may be underestimating the actual new Linux server installations – I believe IDC may be counting only top distributions in their survey.

Client

Starting with Dataquest's market share figures published in June '98, I injected the incremental Linux numbers derived from RedHat's market survey (showing 7.5M users at the end of 1997).

Desktop OS

		1996	1997	growth
Windows NT	units	2,207.70	6,936.70	214%
	%	2.56%	7.09%	
Mac OS	units	3,987.00	2,928.80	-27%
	%	4.63%	2.99%	
OS/2	units	1,832.70	1,053.00	-43%
	%	2.13%	1.08%	
UNIX	units	702.9	770.4	10%
	%	0.82%	0.79%	
Linux	units			
	%			
DOS	units	2,506.70	1,478.90	-41%
	%	2.91%	1.51%	
Proprietary and Others	units	166.8	139.6	-16%
	%	0.19%	0.14%	
Windows 3.1	units	24,508.30	8,066.90	-67%
	%	28.44%	8.25%	
Windows 95	units	50,255.70	76,446.60	52%
	%	58.32%	78.15%	
Totals	units	86,167.80	97,821.00	14%
	%	100	100	

Distributor Market Share

IDC provides information on the relative market share of the Linux distributors:

		1996	1997
Red Hat	units	800,000	1,000,000
	%	51.9%	54.4%
Caldera	units	85,000	123,000
	%	5.5%	6.7%
Workgroup	units	50,000	75,000
	%	3.2%	4.1%
Walnut Creek	units	35,000	40,000
	%	2.3%	2.2%
Others	units	570,000	600,000
	%	37.0%	32.6%
Total	units	1,540,000	1,838,000
	%	100.0%	100.0%

Linux Qualitative Assessment

I purchased and installed a copy of Caldera's OpenLinux v1.2 standard edition. I installed it on an old P5-100 / 32MB RAM machine in my office that used to run NT4. Knowing that device driver support on Linux was well below NT's, I intentionally chose a machine and peripherals that represented the 80% of the *installed base* (e.g. 3c509 NIC, Adaptec SCSI controller, etc.)

Installation

Caldera provided an auto-run CD which launched directly into their setup program – "LISA". Lisa prompted me for:

- Language selection (an interesting future research project would be to truly understand the depth of localization support provided by Linux.)
- Keyboard selection
- IDE hard drive & CD ROM device detection.

Although the dialogs could use a lot of work (e.g. many questions were phrased as double-negatives – "Should setup disable plug & play device detection (yes/no)"), up to this point I was asked no questions that a power user couldn't correctly answer.

A second round of device detection impressively auto-discovered my:

- Adaptec SCSI adapter
- Plextor CD-ROM drive
- Seagate Hard Drive
- 3Com 3c509 Ethernet adapter

I selected default device settings for each hardware option, selected "typical" install options and then LISA started copying.

This phase of the install/setup process was finished in 30 minutes (most of that time copying) and with a total of ~15 dialog boxes.

UI

As mentioned earlier, one of the quirks of UNIX / Linux relative to NT is that video drivers run in userspace and are not required for most system functionality. Linux is quite content with just a command prompt.

A second round of installation scripts was necessary to install the GUI. The installer gave me the option of choosing which video subsystem to install / configure and I chose the Xfree86 server because it's an entirely open source system (the other option – MetroX – was provided by Caldera and is believed to be the more stable codebase).

This part of setup definitely required knowledge of video systems even beyond many power users. Not only did I have to know the name / make / model of my video card and chipsets but I was presented with questions about their revision numbers, the scan rates of my monitors, etc. After significant trial and error, I finally got my video system working correctly.

The latest generation Xfree86 + CDE was slick and definitely represented among the best-of-breed in UNIX GUI's. A SUN desktop user would be perfectly at home here. *An advanced Win32 GUI user would have a short learning cycle to become productive.*

Following UNIX philosophy, however, mastery of the GUI was not enough to use the full system. Simple procedures such as reading a file from a floppy disk required jumping into a terminal window, logging in as administrator, and running an arcane "mount" command.

Networking

A very illustrative case of how the Linux user community works was revealed by my experiences with the networking subsystem.

Caldera's OpenLinux installer only provided the client daemon to handle the BootP protocol (as opposed to DHCP) and for some reasons, it didn't install correctly. I looked around on the CD that Caldera provided for a DHCP daemon and couldn't find one.

A small number of web sites and FAQs later, I found an FTP site with a Linux DHCP client. The DHCP client was developed by an engineer employed by Fore Systems (as evidenced by his email address; I believe, however, that it was developed in his own free time). A second set of documentation/manuals was written for the DHCP client by a hacker in *Hungary* which provided relatively simple instructions on how to install/load the client.

I downloaded & uncompressed the client and typed two simple commands:

Make – compiles the client binaries

Make install –installed the binaries as a Linux Daemon

Typing "DHCPCD" (for DHCP Client Daemon) on the command line triggered the DHCP discovery process and voila, I had IP networking running.

DHCP as an example of Linux process

Since I had just downloaded the DHCP client code, on an impulse I played around a bit. Although the client wasn't as extensible as the DHCP client we are shipping in NT5 (for example, it won't query for arbitrary options & store results), it was obvious how I could write the additional code to implement this functionality. The full client consisted of about 2600 lines of code[3].

One example of esoteric, extended functionality that was clearly patched in by a third party was a set of routines to that would pad the DHCP request with host-specific strings required by Cable Modem / ADSL sites.

A few other steps were required to configure the DHCP client to auto-start and auto-configure my Ethernet interface on boot but these were documented in the client code and in the DHCP documentation from the Hungarian developer.

Key takeaways here:

- Contrary to popular belief, even though this was open source, I never had to touch the 'C' code to get the core functionality working.
- The author of the driver and the author of the documentation were two geographically separated individuals.

[3] By contrast, the NT5 DHCP client consists of 20,000 LOC. RameshV from the NT5 team pointed out some of the incremental functions in our DHCP client: including thread-safety, API's, auto-net support, suspend/resume PNP support, multiple interface support, additional error checking functionality + messages. Notice that these features aren't DHCP features *per se* but rather, integrative/scenario-driven functionality.

- The GPL + incremental improvements process had *already* been at work as evidenced by the Cable Modem/ADSL extensions.

- **Most importantly:** a process that NatBro pointed out in the OSS paper – "a modestly skilled UNIX programmer can grow into doing great things with Linux". I'm a poorly skilled UNIX programmer but it was immediately obvious to me how to incrementally extend the DHCP client code (the *feeling* was exhilarating and addictive).

  Additionally, due directly to GPL + having the full development environment in front of me, I was in a position where I could write up my changes and email them out within a couple of hours (in contrast to how things like this would get done in NT). Engaging in that process would have prepared me for a larger, more ambitious Linux project in the future.

Apps

Caldera bundled StarOffice from Star Corp in Germany. The Office team is quite familiar with StarOffice as a "second-string" contender in the suite category after Corel (which is bundled with Red Hat) and Lotus.

StarOffice was almost entirely an Office 97 clone from a UI perspective. The menus, buttons, placement, etc. were all generally identical. In many cases, large areas of functionality in the menu bar were missing (e.g. Macros). Other stereotypical Office97 features (e.g. red squiggles under misspelled words) were correctly replicated.

As a test, I tried importing a somewhat simple PowerPoint document into StarOffice from a floppy disk. This required jumping into an x-terminal and mounting a new floppy disk into the Linux file system namespace and pointing out to Linux that it was FAT16 formatted. From there, I launched StarOffice's PowerPoint clone and pointed it at the namespace for the floppy and uploaded the file.

Simple slides (such as pure text + bullet points) imported nearly 100% correctly (although fonts and sizing were changed). Complex sides (using PowerPoint's line art, etc.) were almost always totally trashed.

Perceived Performance

Caldera also bundles Netscape's Navigator browser. The browsers UI, of course, perfectly matches Netscape's UI on win32 platforms.

I didn't have the time to run true performance tests, but my anecdotal / percieved performance was impressive. I previously had IE4/NT4 on the same box and by comparison the combination of Linux / Navigator ran at least 30-40% faster when rendering simple HTML + graphics.

Testing end user applications on top of Linux will be an interesting performance test in the future.

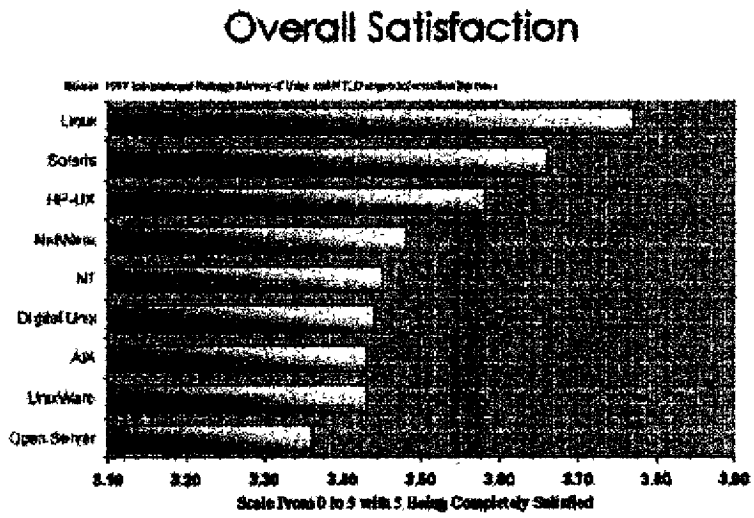
On a negative note, after I had instantiated 3 instances of Navigator on the box, performance came to an almost complete standstill, the mouse become unresponsive, none of the keyboard command sequences worked and I had to reboot the box.

Conclusions

Skilled users with modest developer backgrounds are probably delighted to use Linux due to the endless customizability afforded by Open Source. The simplicity and consistency of the process to modify the system presents a very low learning curve towards "joining" the Linux process.

Long term, my simple experiments do indicate that Linux has a chance at the desktop market but only after massive investments in ease of use and configuration. The average desktop user is unfamiliar with "make".

Linux Competitive Issues



Consumers Love It.

A December 1997 survey of Fortune 1000 IT shops by Datapro asked IT managers to rate their server OS's on the basis of: TCO, Interoperability, Price, Manageability, Flexibility, Availability, Java Support, Functionality, and Performance. RedHat provides summary info at: http://www.redhat.com/redhat/datapro.html.

When overall satisfaction with the OS's was calculated, Linux came out in first place. Linux was rated #1 in 7 of 9 categories in the DataPro study losing only on: functionality breadth, and performance (where it placed #2 after DEC)[4]

[4] George Spix points out that this survey in many cases quotes a self-selected audience. They "must have already been lovers to buy it... all hobbyists like their "rigs".

Linux vs. NT

Windows NT is target #1 for the Linux community. To characterize their animosity towards NT (or, for that matter, anything Microsoft) as religious would be an understatement[5]. Linux's (real and perceived) virtues over Windows NT include:

- **Customization** – The endless customizability of Linux for specific tasks – ranging from GFLOP clustered workstations to 500K RAM installations to dedicated, in-the-closet 486-based DNS servers – makes Linux a very natural choice for "isolated, single-task" servers such as DNS, File, Mail, Web, etc. Strict application and OS componentization coupled with readily exposed internals make Linux ideal.

  The threat here is even more pronounced as over time, the number of servers (and consequently dedication to specific tasks) will increase. Customers enjoy the simpler debugging and fault isolation of individual servers vs a monolithic server runing multiple services.

- **Availability/Reliability** – There are hundreds of stories on the web of Linux installations that have been in continuous production for over a year. Stability more than almost any other feature is the #1 goal of the Linux development community (and the #1 cited weakness of Windows)

- **Scaleability/Performance** – Linux is considered faster than NT in networking, and processes. In particular, as a server, Linux's modular architecture allows the administrator to turn off graphics, and other non-related subsystems for extreme performance in a particular service

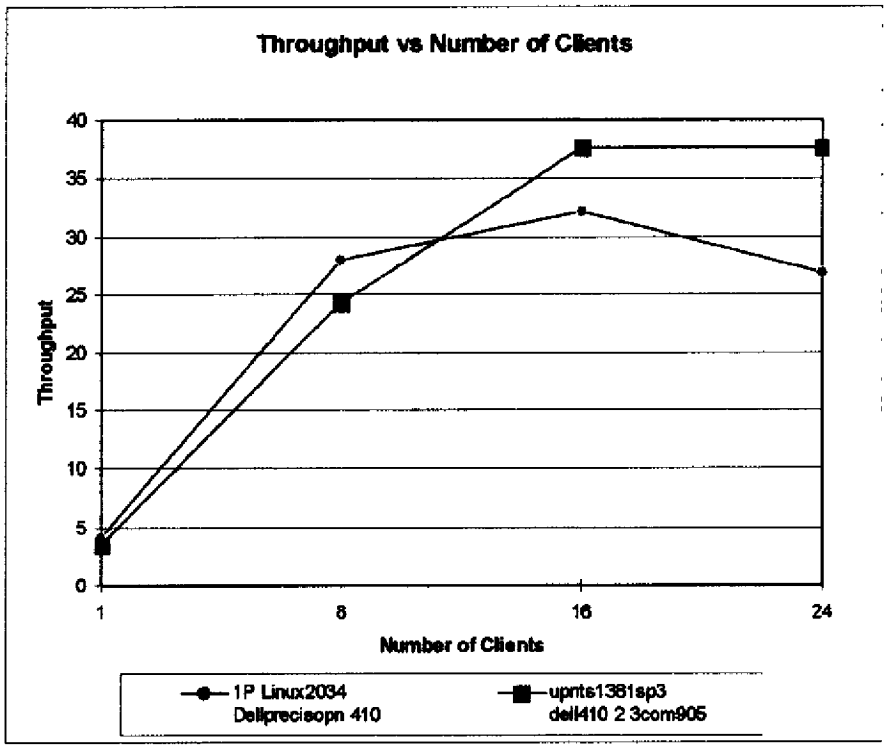
- **Interoperability** – Every open protocol on the planet (and many of the closed ones) have been ported to Linux. In a Windows environment, work from the SAMBA team enables Linux to look like an NT Domain Controller / File Server.

Recently, the NT performance team ran their NetBench file/print test against a recent Linux distribution. Results indicate that although NT slightly outperforms Linux, Linux's performance is still quite acceptable and competitive considering the years of tuning that has been applied to the NT SMB stack.

---

[5] Linus Torvalds himself is not quite as religious as many of the other Linux developers but has some great comments in an interview he gave for Boot magazine:

http://www.bootnet.com/yousaidwhat/lip_linux_manifesto.html.



Linux vs. Java

Linux developers are generally wary of Sun's Java. Most of the skepticism towards Java stems directly from Sun's tight control over the language -- and lack of OSS.

The Linux community has been asking Sun to treat the Linux platform as a tier-1 Java platform almost since the dawn of the language. However, Sun does NOT support the JDK for Linux.

Interestingly, in order to develop the Linux JDK, several Linux developers signed NDA's to develop the port (http://www.blackdown.org). These pressures have also spawned several OSS JVM clones including http://www.kaffe.org.

Linus comments in (http://www.linuxresources.com/news/linux-expo.html)

> While Linus would like to see an officially supported Java Development Kit from Sun, he is still not impressed with Java and would prefer to stay out of the Microsoft/Sun clash over Java purity:

Linux vs. SunOS/Solaris

The Linux community has ambivalent feelings towards Sun. On the one hand, as the most vocal critic of Microsoft, Sun is praised. On the other hand, as the most visible yardstick in the UNIX world, beating Solaris / SunOs is a favorite past-time of Linux hackers.

Using the Lmbench OS benchmark, Linux outperforms SunOS not only on x86 but, impressively, *on Sun Hardware as well* in networking, process / context switch times, disk I/O, etc.

Some (not very scientific or comprehensive) OS performance statistics can be found on: http://www.caip.rutgers.edu/~davem/scoreboard.html.

In generating these performance results, the great number of eyes (and consequently large amount of hand tuning / optimizing of critical code paths) is most frequently cited.

A general architectural comparison citing the performance benefits of Linux over SunOS can be found on: http://www.nuclecu.unam.mx/~miguel/uselinux/SparcLinux.html

Sun has recently announced (8/10) the free licensing of Solaris binaries for non-commercial institutions (http://www.sun.com/edu/solaris/index.html). Presumably this is due to competitive pressures from Linux.

Linux on the Server

The vast majority of Linux's installed, production base is projected to be in servers.

Reasons why Linux is strong in this market include:

- **Unix heritage** – the server market, especially at the high-end, is already familiar & comfortable with UNIX, Internet-based freeware, etc.
- **Professional users** – high end server administrators are often developers/power users themselves and are therefore comfortable with recompiling apps, etc.
- **"Generic" services** – these are services defined via open, lowest-common-denominator protocols such as DNS, SMTP, etc. Functional differentiation is lower in the server market than it is in the client market. There is a lower bar for experimentation with servers since it disrupts downstream client activity very little.
- **Dedicated Functionality** – because servers are typically tasked with a single function (e.g. mail, file/print, database, etc.), the level of required integration with other services and devices in the organization is much lower.

Network infrastructure

Linux is often used to provide commodity, low horsepower, high reliability network infrastructure services. For example:

- DNS
- DHCP

- Print Servers
- File Servers

ISP Adoption

One of Linux's core user bases is ISP shops. Some of the reasons for this include:

- **Cost** – ISP's live on horribly tiny margins. Linux's free price + wide hardware support is consequently very attractive.
- **Maintainability** – If something breaks, it needs to be fixed immediately. In larger ISPs, the technical expertise to debug code breaks or at least install quickly available patches is plentiful. Remote manageability in particular is a key attribute.
- **Reliability** – perception that non-Linux OS's aren't reliable or scalable enough (in particular Windows NT)
- **UNIX background** – ISPs are traditionally Unix havens. ISP sys admins are very well versed in arcane UNIX command line admin, remote administration, etc. In a group that's very predisposed towards UNIX's strengths, Linux represents a best-of-breed UNIX.

Thin Servers

Linux is emerging as a key operating system in the nascent thin server market:

- **Source code availability** – Freely available source code provides for easy customization of the OS
- **Commodity protocols** – Thin servers speak very simple, non-extensible, commodity protocols to clients such as HTTP, SMTP, and SMB.
- **Modularity & Small size** – Because the OS was designed in a very non-integrated, componentized manner from the outset, it's very easy to build boxes that don't have a monitor, keyboard, etc.
- **Cost** – Obviously, margins are very low in embedded devices & a free OS helps
- **Code Maintenance** – Because the Linux source is constantly being upgraded, embedded developers are reassured that new changes / fixes can be snapped back to their systems at any time.
- **Tool Availability** –Unix tools are far more powerful than the current crop of embedded development tools.

One of the most prominent thin-server on the market based on Linux today is the Cobalt Microserver (http://www.cobaltmicro.com). Other thin server vendors (most notably Whistle Interjet) are using FreeBSD derived products.

Case Study: Cisco Systems, Inc.

IDC published a study of 3 corporate IS departments which had significantly deployed Linux. Cisco has several hundred Linux servers deployed through their organization serving the following functions:

- NFS/SMB server
- Print Server (LPD & SMB)
- Small office productivity (ApplixWare office suite, Netscape Navigator)
- WWW Server & Proxy
- Software development

Linux on the Client

Due to it's UNIX heritage and Hacker OS background, Linux is a weak client-desktop OS. Additionally, the OSS paper points out why, in a broad sense, OSS is much more of a server threat than a desktop threat.

There are, however, several initiatives attempting to push Linux as a viable desktop replacement[6]. Each of the various Desktop environments (GNOME, KDE, CDE) come bundled with basic productivity applications and there are 2 full fledged office suite products (from Corel and StarOffice) which provide varying degrees of file format compatibility with Microsoft Office.

App / GUI Chaos

Unlike the Kernel – where Linus Torvalds maintains the core source tree, the Linux GUI has NOT been singularly managed and consequently has a highly forked tree.

Linux does not have a consistent UI look and feel due to the variety of widget sets (a widget is analogous to an OCX or VBX) such as Motif, LessTiff, MIT Athena, Sun OpenLook, etc. Because widgets represent central UI concepts (such as a close button, dropdown menu, dialog box, etc.), users get different look-and-feels and often different usage semantics.

In addition to Widgets, the "desktop" or "shell" has also forked. Primary players in the shell arena include:

- **Common Desktop Environment (CDE)** – a collaboration between major commercial Unix manufacturers. CDE, however, is not GPL'd and has thus resulted in multiple Linux groups creating CDE replacements. CDE is available on Linux.
- **K Desktop Environment (KDE)** - a "free" CDE clone. KDE replaces all functionality in CDE but does not provide a widget set. (in practice the widget set is actually

[6] Linus Torvalds states that his intent has always been to target Linux at the desktop (http://www.heise.de/ct/english/98/16/032/): "Note that the reason I propagate Linux as a desktop operating system is because I think that's the more difficult market. I was personally never very worried about Linux as a server platform - servers are "easy" compared to desktops, because in the end servers are fairly anonymous - you only see the network behaviour of a server, while with a desktop system it's much more of a "complete immersion" environment, not just the network part"

MORE lines of code than the desktop). Consequently, the KDE developers chose the QT widget set which was most liberally licensed – but still not GPL'd – and compatible with most other UNIX systems. This however, launched the final band of GPL zealots who created…

- **Gnome** – a radical new UI initiative based loosely on X-windows and incorporating CORBA into the desktop. While this is an ambitious task, and may be more revolutionary than CDE is, its long from being complete and is lacking in application support.

The lack of singular, customer-focused management has resulted in the unwillingness to compromise between the different initiatives and is evident of the management costs in the Linux process.

Unix Developers

Linux as a client has found a home with UNIX developers. Many developers prefer to use Linux for their dev machines in order to write code for other UNIX platforms. The ease of debugging on top of a platform where there is open source is often cited.

Non-PC Devices

Corel's NC devices were based on a Linux derived OS. These efforts, however, have since been suspended (with the Corel developed application-level code being returned to the OSS community)

Linux Forecasts & Futures

Current Initiatives / Linux Futures

There are literally hundreds of small research projects attempting to improve various parts of the Linux OS.

Some projects include:

- **Linux 2.2** – High Availability features such as deeper RAID support (RAID 0, 1, 5 supported today), volume management; file system performance improvements; asynchronous I/O & completion ports; Ipv6; . An excellent feature summary can be found on: http://wn.net/980730/a/2.2chFinal.html.

- **Linux 3.0** – Linus forecasts that the next version of the Kernel will incorporate better SMP scalability and begin to attack the clustering problem. Development is far from starting so details / commitments are extremely sketchy.

- **Beowulf clustering** – Beowulf is a shared-nothing cluster that runs today on Linux. It requires specially developed applications which are able to spawn subprocesses on remote hosts for computing. As such, it is not a real competitor to WolfPack and most of the magic in Beowulf in the applications rather than system services. However, as a press-magnet, Beowulf clusters with appropriate software have been demonstrated at supercomputer power (a 10GFLOP was recently ranked #315 on the top 500 supercomputers list maintained by the NCSA).

- **DIPC** – Distributed Inter-Process Control Pack – provides standard IPC functions to client apps (semaphores, shared memory, etc.) but is able to remote those functions to network hosts.

- **GNOME** – Next generation UI initiative for Linux loosely based on X-windows +CORBA . More info at http://www.gnome.org. Many of the key developers for Gnome work for RedHat.

"Parity Growth"

The biggest future issue for Linux is what to do once they've reached parity with UNIX. JimAll used the phrase "chasing taillights" to captures the core issue: in the fog of the market place, you can move faster by being "number 2 gaining on number 1" than by being number 1.

Linux has now reached parity / incrementally ahead of other Unixes. Consequently, it will be much harder to achieve the big leaps the development team is accustomed to.

From Wired's piece on Linux:

> *This two-track development process has made Linux probably more advanced and yet more stable than any other version of Unix today. "Linux is now entering an era of pure development instead of just catching up," says Jacques Gélinas.*

Strengths

A second paper on "Open Source Software" goes into depth on the generic advantages of the Open Source Process.

Unix Heritage & Fast Copying

Linux unabashedly steals the best ideas from the various UNIX flavors. This means free R&D. Recently, Linux has begun to copy NT-ish features such as transmitfile(), a hacked form of IO Completion Ports, etc.

Established / high-visibility bazaar

Linux is the most often cited example of a "credible" open source project. By being the largest OSS project today, it's the most sustainable in the future.

Dominance in Education / Research Markets

New ideas from academia + new computer scientists are being trained wholesale in the Linux OS. In particular, Europe and Asia are very hooked on the Linux OS. Email from BartelB (Marketing Manager EdCU):

> For higher education in particular, Linux represents an alternative to the Commercial demons of software, (not a quantitative statement but in talking with many CS students who supply 60% of the labor for higher education IT departments, they have express these feelings and its a problem). They feel that once they commit to a windows platform there creativity will be lost. Money is not there driving force, they don't want to be "Borged".

Weaknesses

The paper on "Open Source Software" provides general process weaknesses. Here, we'll try to list only the weaknesses that are unique to Linux.

Unix Heritage

Linux's biggest advantage can also quickly become a disadvantage – particularly in volume markets where ease of use is paramount. Some nascent efforts have been launched to make Linux friendlier but they are generally receiving relative apathy from the dev community (http://www.seul.org).

Too Many Managers

In a typical Linux distribution, the majority of the code comes from sources outside of the main Linux tree. This piecemeal approach will make it especially hard to solve architectural problems and launch new, cross-component initiatives.

Worst case scenarios

This section is pure speculation. What are some of the worst case scenarios for Linux to hurt Microsoft?

Customer Adoption – It gets good enough

Using today's server requirements, Linux is a credible alternative to commercial developed servers in many, high volume applications. The effect of this on our server revenue model would be immense.

Our client-side revenue model is still strong however for a variety of reasons including switching costs for the entire pool of win32 source code. Linux advocates, however, are working on various emulators and function call impersonators to attack this cost.

This points back to an obvious solution – innovation in the core platform is an ongoing requirement.

Channel Adoption

The "Open Source Software" paper has a section on OSS business models. Summarizing that section, there are 4 primary business models we have identified for Open Source Software.

1. **Secondary Services** – The vendor / developer of OSS makes their money on service contracts, customer integration, etc.

2. **Loss Leader for Market Entry** – The vendor / developer of OSS uses OSS's process advantages (in particular credibility) as a lever against established commercial vendors.

3. **Commoditizing Downstream Suppliers** – The vendor / developer of OSS is also the producer of a product / service further in the value chain and closer to the consumer.

4. **Standards Preemption** – Because OSS process are argued to be winner-take-all, it may suit the vendor / developer to seed the OSS market with their codebase to pre-empt a competitive codebase from taking hold.

IBM Adopts Linux?

IBM is most capable at capturing revenues from all 4 of the business models associated with Linux.

1. **Secondary Services** – IBM is very strong in consulting, integration, support, etc. This is their fastest growing business today

2. **Loss Leader** – IBM's client/low-end operating system business is in shambles (remember OS/2?). Additionally, IBM has stumbled on various NC/JavaOS systems as well. By leveraging Linux's credibility (as well as applying IBM's development resources toward improving ease of use?), IBM would hope to upset the status quo in the volume OS space and hope to capture revenue in the ensuing disruption.

3. **Commoditizing Downstream Suppliers** – As a PC/Hardware OEM, IBM's margins increase by commoditizing a key cost item – the OS. In particular, the commoditized & highly customizable qualities of the Linux OS actually provide greater differentiation for hardware vendors.

4. **Standards Preemption** – The standard to pre-empt is anything Microsoft – in particular new OS services that we integrate directly into future versions of NT.

IBM, despite their Apache announcements, seems unlikely to advocate this in the short run. I'd imagine that religion within their various OS development efforts alone would provide a significant amount of near term inertia.

Sun Adopts?

Sun's rationale for adopting Linux would be less encompassing than IBM's.

1. **Secondary Services** – Sun is not very strong in consulting / integration revenue. They do, however, make significant revneue in support and mainrtenance.

2. **Loss Leader** – Sun could market Linux as a low-end OS and try to make money in the UNIX applications space above it. Because Linux could potentially be a far larger market than anything Sun is accustomed to, this would be a net positive for them.

3. **Commoditizing Downstream Suppliers** – Sun is also a hardware vendor (with some excellent systems). Sun would lose their current OS revenue but the ability to sell their hardware into a broader channel could be compelling.

   Linux adoption, however, puts Sun at significant risk if their SPARC operations cannot keep up with Intel's innovation pace.

4. **Standards Preemption** – Beat Microsoft Standards

PC OEM's

Other worst case adoption scenarios are subsets of the Sun / IBM case and involve other PC vendors such as Compaq and Dell.

Note, however, that Compaq and Dell merely have to credibly threaten Linux adoption in order to push for lower OEM OS pricing.

Server ISVs

One interesting spin on the "Commoditizing Downstream Suppliers" strategy could be **backward integration** by server ISV's. For example, Oracle could ship a version of Parallel Server for Linux that *includes the Linux OS within the distribution.*

This is basically a play on the thin-server concept. Instead of *integrating* multiple small business functions on a single server, this attempts to *disintegrate* the features of an enterprise OS into the minimal set necessary to run the specific server application. It plays into the business models identified as follows:

1. **Secondary Services** – Companies like Oracle/SAP/Baan/etc. already make a large percentage of their income from on-site consulting agreements
2. **Loss Leader** – treating the OS as a loss leader helps them concentrate revenues for a particular hardware unit into their hands
3. **Downstream commoditization** – Oracle has no problem declaring the Server OS as a kernel, memory manager, IP stack, and some disk.
4. **Standards Preemption** – beat Microsoft.

Next Steps & Microsoft Response

A lot more thought and work needs to go into formulating Microsoft's response to Linux. Some initial thoughts on how to compete with Linux in particular are contained below. One "blue sky" avenue that should be investigated is if there is any way to turn Linux into an opportunity for Microsoft.

A more generalized assessment of how to beat the Open Source Software process which begat Linux is contained in the "Open Source Software" document.

Beating Linux

Beat UNIX

The single biggest contributor to Linux's success is the general viability of the UNIX market. Systematically attacking UNIX in general helps attack Linux in particular. Some Linux-targeted initiatives in this space (not a comprehensive list) include:

- **Improve Low-End "IAM"** – Scaleability, Interoperability, Availability, and Management (SIAM) are the most often cited reasons for using UNIX over NT in mission critical, high-end applications.

In today's Linux deployments however, scaleability is not the driver as much as Interop, Reliability, and Headless Management.

- **UNIX services for NT Add-on pack**

Modularize / Embed Windows NT

Relative to other UNIX's Linux is considered more *customizable*. Addressing this functionality involves more than just the embedded Windows NT project. Greater componentatization & general dependency reduction within NT will improve not only it's stability but also the ability of highly skilled users/admins to deploy task-specific NT installations.

This requires:

- Wide availability of the Embedded NT toolkit
- Greater focus on ease-of-use in the toolkit

Beat commodity protocols / services

Linux's homebase is currently commodity network and server infrastructure. By folding extended functionality into today's commodity services and create new protocols, we raise the bar & change the rules of the game.

Some of the specifics mentioned in the OSS paper:

- **DNS integration with Directory**. Leveraging the Directory Service to add value to DNS via dynamic updates, security, authentication
- **HTTP-DAV**. DAV is complex and the protocol spec provides an infinite level of implementation complexity for various applications (e.g. the design for Exchange over DAV is good but certainly not the single obvious design). Apache will be hard pressed to pick and choose the correct first areas of DAV to implement.
- **Structured storage**. Changes the rules of the game in the file serving space (a key Linux/Apache application). Create a compelling client-side advantage which can be extended to the server as well (e.g. heterogenous join of client & server datastores).
- **MSMQ for Distributed Applications**. MSMQ is a great example of a distributed technology where most of the value is in the services and implementation and NOT in the wire protocol.

Leverage ISV's for system improvements

A key long term advantage that Linux will enjoy is the massive pool of developers willing to improve areas of the core platform. Microsoft will never be able to employ a similar headcount.

A key mechanism to combat this is to make it easy (and provide incentives) for ISV's to extend system components in NT for custom, vertical applications. One example here could be Veritas' specialized file system drivers for NT.

"WinTone"

Linux's modularity and customization also implies inconsistencies in services available on an arbitrary Linux installation. Microsoft can provide a bundle of services that are universally available in all OS releases (current initiatives include WBEM-based management) that generate network externalities when combined across many devices in the network.

Put another way, the extreme modularity of Linux devalues what a "Linux-logo'ed" app means. By contrast, Window's monolithic nature gives an app developer more leeway in terms of what API's are callable.

Process Vulnerabilities

Where is Microsoft vulnerable to Linux? As stated earlier, the primary threat resides on the server vs. the client.

Linux will "Cream Skim" the Best NT Server Features

The Linux community is very willing to copy features from other OS's if it will serve their needs. Consequently, there is the very real long term threat that as MS expends the development dollars to create a bevy of new features in NT, Linux will simply cherry pick the best features an incorporate them into their codebase.

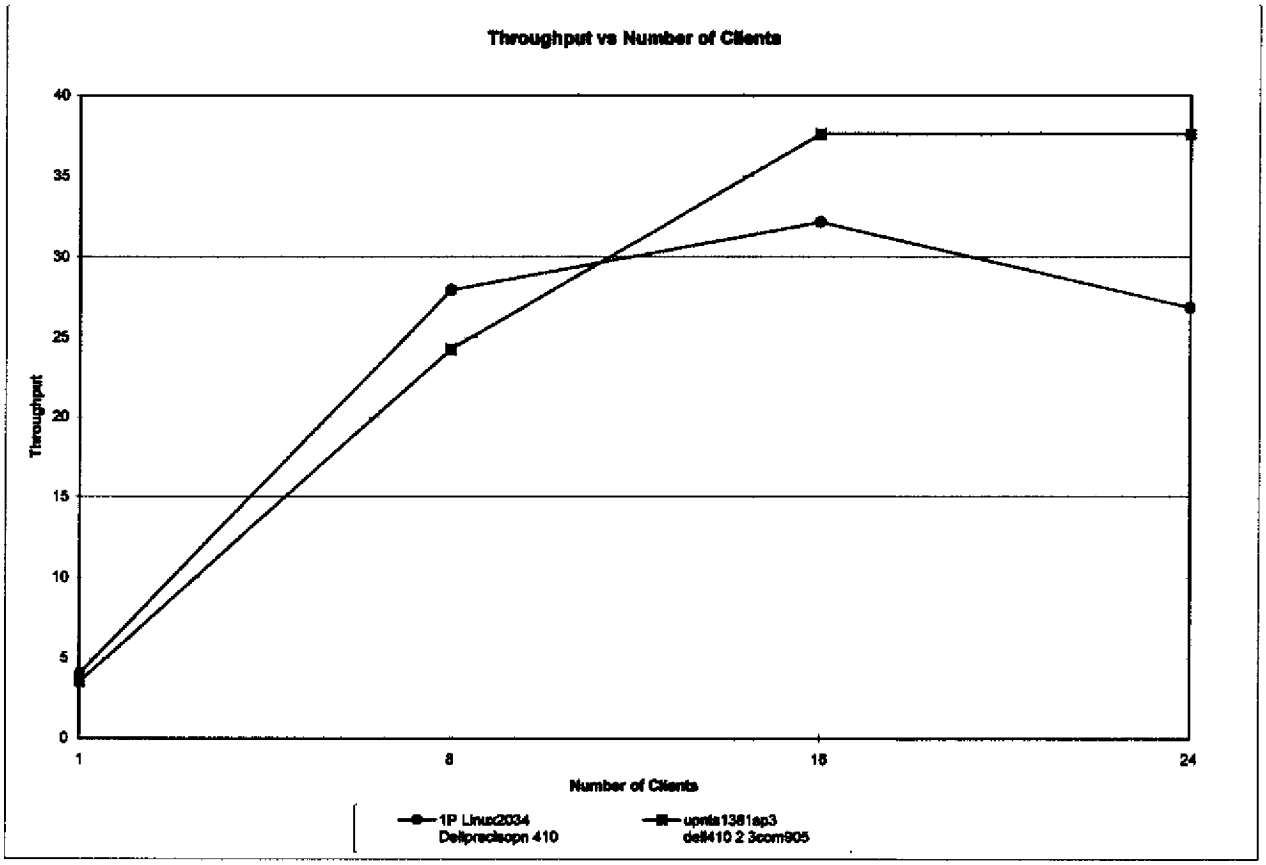
The effect of patents and copyright in combatting Linux remains to be investigated.

Linux is recreating the MS "3rd release is a charm" advantage – FASTER

Microsoft's market power doesn't stem from products as much as it does from our iterative process. The first release of a Microsoft product often fairs poorly in the market and primarily of generates fine granularity feedback from consumers. Similarly, Linux has shown that they are capable of iterative cycles – but at an order of magnitude faster rate. On the flip side, however, our incremental releases are arguably much larger whereas many of Linux's incremental releases are tantamount to pure bug fixing.

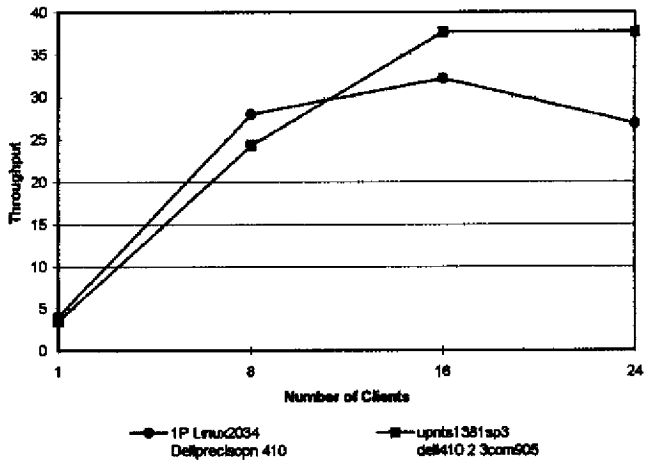
Revision History

Date	Revision	Comments
8/03/98	0.95	
8/10/98	0.97	Started revision table Added reference to SUN's non-commercial license Added Linus quote for desktop vs. server issues
8/11/1998	1.00	Added perf charts, published first release.


Throughput vs Number of Clients
Throughput
40
35
30
25
20
15
10
5
0
1
8
16
24
Number of Clients
1P Linux2034
Dellprecisopn 410
upnta1381ap3
dell410 2 3com905

Overall Results

dm_8_clients	2	8	10967926.717	83.831	1430669.450	8	1324460.000	1	27.94
dm_24_clients	4	24	10559994.514	80.566	469185.020	15	405406.477	8	26.66



Used 1-72-8 script toned down to 16 clients,
Used two subnets

User = 4.6 %
System = 13.2 %
Idle = 82.2 %

Suite Definition

dm_8_clients	2	8	N/A	2.000	5.000	Disk Mix	N/A	N/A	N/A	Iterations	1 000	1 000	3.000
dm_24_clients	4	24	N/A	2.000	5.000	Disk Mix	N/A	N/A	N/A	Iterations	1 000	1.000	3.000

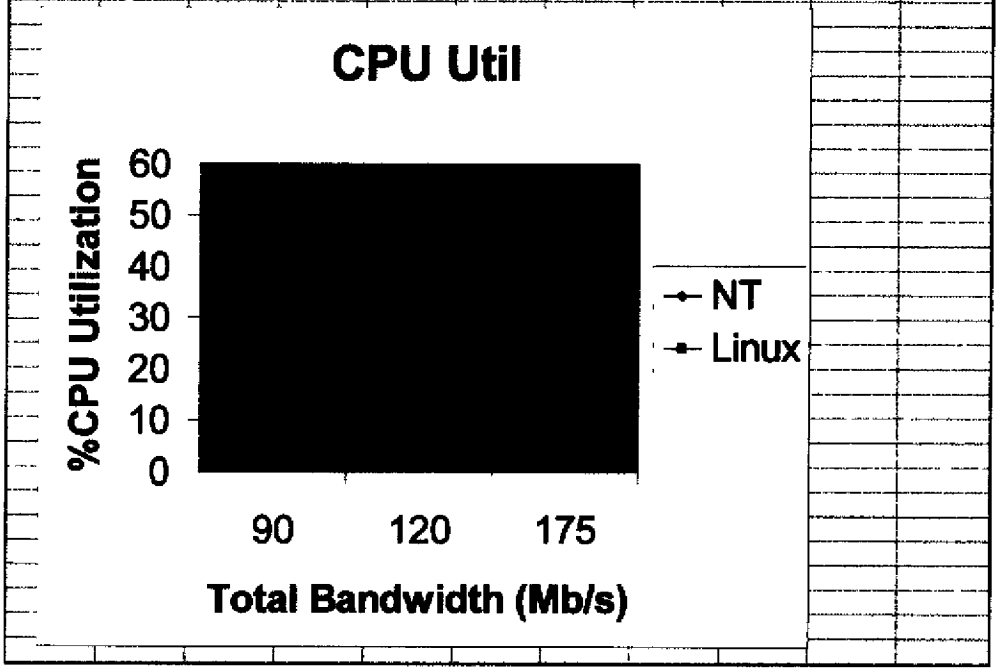
dm_8_clients	2	1	f:\clients\client1	f:\clients\client1
		3	f:\clients\client3	f:\clients\client3
		5	f:\clients\client5	f:\clients\client5
		7	f:\clients\client7	f:\clients\client7
dm_16_clients	3	1	f:\clients\client1	f:\clients\client1
		3	f:\clients\client3	f:\clients\client3
		5	f:\clients\client5	f:\clients\client5
		7	f:\clients\client7	f:\clients\client7
		9	f:\clients\client9	f:\clients\client9
		11	f:\clients\client11	f:\clients\client11
		13	f:\clients\client13	f:\clients\client13
		15	f:\clients\client15	f:\clients\client15
dm_24_clients	4	1	f:\clients\client1	f:\clients\client1
		3	f:\clients\client3	f:\clients\client3
		5	f:\clients\client5	f:\clients\client5
		7	f:\clients\client7	f:\clients\client7
		9	f:\clients\client9	f:\clients\client9
		11	f:\clients\client11	f:\clients\client11
		13	f:\clients\client13	f:\clients\client13
		15	f:\clients\client15	f:\clients\client15
		17	f:\clients\client17	f:\clients\client17
		19	f:\clients\client19	f:\clients\client19
		21	f:\clients\client21	f:\clients\client21
		23	f:\clients\client23	f:\clients\client23

Client Data

dm_8_clients	2	client1	1	1	74699544	56.400	1	1324460.000
		client3	3	3	74699544	56.042	1	1332920.738
		client5	5	1	74699544	53.290	1	1401765.376
		client7	7	3	74699544	52.819	1	1419630.628
dm_16_clients	3	client1	1	1	74699544	93.113	1	802246.131
		client3	3	3	74699544	95.067	1	785756.824
		client5	5	1	74699544	93.443	1	799412.947
		client7	7	3	74699544	94.056	1	794202.658
		client9	9	1	74699544	101.897	1	733068.748
		client11	11	3	74699544	96.517	1	773962.186
		client13	13	1	74699544	91.619	1	815328.087
		client15	15	3	74699544	90.714	1	823462.134
dm_24_clients	4	client1	1	1	74699544	178.279	1	419003.607
		client3	3	3	74699544	166.120	1	449672.189
		client5	5	1	74699544	175.609	1	425228.948
		client7	7	3	74699544	161.093	1	463704.489
		client9	9	1	74699544	174.918	1	427054.643
		client11	11	3	74699544	181.184	1	412285.544
		client13	13	1	74699544	158.115	1	472436.061
		client15	15	3	74699544	149.643	1	499185.020
		client17	17	1	74699544	176.468	1	423303.624
		client19	19	3	74699544	156.074	1	478616.195
		client21	21	1	74699544	160.413	1	465670.139
		client23	23	3	74699544	174.473	1	428143.882

dm_8_clients	2	client1	1324460.000	-1.282	
		client3	1332920.738	-1.081	
		client5	1401755.378	0.739	
		client7	1419630.628	1.207	
dm_16_clients	3	client1	802246.131	0.404	
		client3	785756.824	-0.154	
		client5	799412.947	0.308	
		client7	794202.658	0.132	
		client9	733088.748	-1.936	
		client11	773952.196	-0.553	
		client13	816328.067	0.847	
		client15	823462.134	1.122	
dm_24_clients	4	client1	419003.507	-0.869	
		client3	449672.189	0.396	
		client5	425226.946	-0.604	
		client7	463704.469	0.870	
		client9	427054.643	-0.530	
		client11	412286.544	-1.134	
		client13	472438.061	1.327	
		client15	499186.020	2.421	MAX
		client17	423303.624	-0.683	
		client19	478816.195	1.580	
		client21	465970.139	1.050	
		client23	428143.862	-0.495	

1 Pentium II 400mhz Linux Dell vs Intel 4way Xeon										
		1NIC	N1	N2	N3	Total		NT CPU	LX CPU	%diff
1NIC	AMD		90			90		22	12	54.55%
2NIC	3COM		60	60		120		35	20	57.14%
2NIC	AMD		87	88		175		50	40	80.00%



[redacted]

Server Name:
Version:
CPU/Memory:
Processor Speed:
Memory:
Network Configuration:
Comments:

client2	2	Intel Pentium(R) Step 5 Features 1bfh, 100 MHz, mc:On-Chip	MS-DOS 7.10
client4	4	Intel Pentium(R) Step 5 Features 1bfh, 100 MHz, mc:On-Chip	MS-DOS 7.10
client6	6	Intel Pentium(R) Step 5 Features 1bfh, 100 MHz, mc:On-Chip	MS-DOS 7.10
client8	8	Intel Pentium(R) Step 5 Features 1bfh, 100 MHz, mc:On-Chip	MS-DOS 7.10
client10	10	Intel Pentium(R) Step 5 Features 1bfh, 100 MHz, mc:On-Chip	MS-DOS 7.10
client12	12	Intel Pentium(R) Step 5 Features 1bfh, 100 MHz, mc:On-Chip	MS-DOS 7.10
client14	14	Intel Pentium(R) Step 5 Features 1bfh, 100 MHz, mc:On-Chip	MS-DOS 7.10
client16	16	Intel Pentium(R) Step 5 Features 1bfh, 100 MHz, mc:On-Chip	MS-DOS 7.10
client18	18	Intel Pentium(R) Step 5 Features 1bfh, 100 MHz, mc:On-Chip	MS-DOS 7.10
client20	20	Intel Pentium(R) Step 5 Features 1bfh, 100 MHz, mc:On-Chip	MS-DOS 7.10
client22	22	Intel Pentium(R) Step 5 Features 1bfh, 100 MHz, mc:On-Chip	MS-DOS 7.10
client24	24	Intel Pentium(R) Step 5 Features 1bfh, 100 MHz, mc:On-Chip	MS-DOS 7.10

[redacted]

Microsoft Windows 95	32MB	n/a	zero	n/a	n/a	n/a	n/a
[redacted]							
Microsoft Windows 95	32MB	n/a	zero	n/a	n/a	n/a	n/a
[redacted]							
Microsoft Windows 95	32MB	n/a	zero	n/a	n/a	n/a	n/a
[redacted]							
Microsoft Windows 95	32MB	n/a	zero	n/a	n/a	n/a	n/a
[redacted]							
Microsoft Windows 95	32MB	n/a	zero	n/a	n/a	n/a	n/a
[redacted]							
Microsoft Windows 95	32MB	n/a	zero	n/a	n/a	n/a	n/a
[redacted]							
Microsoft Windows 95	32MB	n/a	zero	n/a	n/a	n/a	n/a
[redacted]							
Microsoft Windows 95	32MB	n/a	zero	n/a	n/a	n/a	n/a
[redacted]							
Microsoft Windows 95	32MB	n/a	zero	n/a	n/a	n/a	n/a
[redacted]							
Microsoft Windows 95	32MB	n/a	zero	n/a	n/a	n/a	n/a
[redacted]							
Microsoft Windows 95	32MB	n/a	zero	n/a	n/a	n/a	n/a
[redacted]							
Microsoft Windows 95	32MB	n/a	zero	n/a	n/a	n/a	n/a