From: Carl Stork (Exchange)
Sent: Tuesday, May 06, 1997 7:33 PM
To: Bill Gates; Bill Veghte; Marshall Brumer; Ed Stubbs; Bill Krueger; Forrest Foltz
Cc: Aaron Contorer; Mike Porter; Moshe Dunie; Jim Allchin (Exchange); Bill Krueger; Jon DeVaan; Steven Sinofsky
Subject: RE: Applications boot time

I won't speak to how we do DLL loading or can improve it - Ed, Bill or Forrest should jump in here.

Regarding the license, Intel's original proposal was to tie the use of their algorithm to only Intel-manufactured MMX CPUs. We got them to back off from this completely so we can use their algorithm on both MMX and non-MMX and both Intel and non-Intel x86 systems, in both the OEM product and the retail upgrade. We'll see if we can structure the license such that we can have complete residuals to the algorithm so we have flexibility to figure out if Office Setup can benefit from this, though I don't know how it would control its sector layout. Once we have Intel's algorithm we should certainly show it to the Office team. We asked to get rights for Windows NT but Intel was not willing to give us that license without an obligation on our part to implement it on both NT4 and NT5. Given the shortness of time and the fact that we haven't even evaluated the viability, we left it at MS & Intel will negotiate in good faith to license the technology for Windows NT. I expect that we will get full residuals and not get polluted by working with their algorithm.

Here is some additional info from Bill Krueger:

- The portion of the optimization that intel wrote and that we're using is a DLL that, in and of itself, does nothing to optimize the disk. It takes as input a series of numbers passed in an in-memory data structure, and returns as output the same series of numbers in a reordered in-memory list. It's main task is principally number crunching.
- Defrag uses this information to do the reordering on the set of files that are contained in this list. So defrag is an integral part of this tool: it actually does the placement of the file fragments, where possible, in the order specified.
- The app logging vxd actually collects the file access information in the first place, the intel DLL does not gather any data. Both the app logger and defrag rely on functionality that is available only in the mephis versions of the ifs manager and vfat.
- This set of components could be installed on OSR2 and the optimization would be available to OSR2 customers. However, the same is not true of win95.

-----Original Message-----
From: Bill Gates
Sent: Tuesday, May 06, 1997 7:06 PM
To: Carl Stork (Exchange); Bill Veghte; Marshall Brumer
Cc: Aaron Contorer; Mike Porter; Moshe Dunie; Jim Allchin (Exchange); Bill Krueger; Jon DeVaan; Steven Sinofsky
Subject: RE: Applications boot time

I think we have done a poor job getting very smart people to think about how the whole DLL load process makes Windows slow. The whole way DLL look up is done, the whole way DLL loading is done is very poor.

I think there are the following action items:

a) Windows. Apply IQ to DLL loading. How can it be improved. How should applications developers use it. How should the file system optimize placements better in the first place without requiring a utility to be run.
b) Intel code. We should go ahead and get the license and give them their attribution. However over time I want us to have most of this optimization take place by design or in the backround rather than by having the user have to run something complex and time consuming. I assume the Intel deal lets us sell Memphis upgrades to any chip even if the OEM license somehow restricts something to MMX machines. I assume we are not restricted from doing optimizations ourselves in the future in an unrestricted way. I assume the Intel code will be put on x86 NT as well as Memphis. I assume we don't get "polluted" by understanding the Intel alogortihms.
c) Office. It must be possible for Office setup to be a lot smarter and effectively get most of the speed up that running this Intel utility provides. I am not saying Office has to use the Intel code. Unfortunately the Office group didn't come up with a way to highlight MMX so Intel won't be helping us with Office so we are probably on our own.

I have to say given the importance of speeding up boot time for Office and how important this is for Windows I expect Office to get involved in understanding these speedups. I expect the optimal DLL loading strategy to be well understood.

-----Original Message-----
From: Carl Stork (Exchange)
Sent: Monday, May 05, 1997 5:59 PM

To: Bill Veghte; Bill Gates; Marshall Brumer
Cc: Aaron Contorer; Mike Porter; Moshe Dunie; Jim Allchin (Exchange); Bill Krueger
Subject: RE: Applications boot time

To reinforce a couple of points on the Intel app launch acceleration utility.

1) I don't think that Intel would agree to an Office-only deal with Microsoft, or even a Memphis/NT5 & Office deal. If Intel gives us rights to ship this with Office, they'll want to give a license to Corel and to Lotus as well.

2) The user experience of running the app defrag as part of Office is not great. It definitely takes a while to run. I don't think you would want to do it as an automatic part of Office setup, it would add significant time to setup. What the utility does is dependent on the profile of the individual disk, so Office would not ship pre-configured for this load acceleration. So you then have a utility to run at some later time. Would you allow users to also optimize load times for other apps, or just the office apps? Would you want to replace the disk defrag at the same time? Etc.

Intel is willing to give us the utility on an exclusive basis for MS operating systems, meaning they would not license it to OEMs for inclusion in Win95, they would not license it to utility vendors, etc. the main thing Intel wants in return is attribution.

-----Original Message-----
From: Bill Veghte
Sent: Saturday, May 03, 1997 10:15 AM
To: Bill Gates; Marshall Brumer
Cc: Aaron Contorer; Carl Stork (Exchange); Mike Porter; Moshe Dunie; Jim Allchin (Exchange); Bill Krueger
Subject: RE: Applications boot time

We are pursuing aggressively for Memphis. The question we need to think about is what the coordinated strategy across MS you want us to pursue on the Intel work. We are combating a bloatware perception w/ Office and reducing Office load time would help diminish this perception. At the same time, the Office business is an incredible revenue producer so getting this added "benefit" could help. However, we need to factor in a couple of other elements as well tho as we think about this. They are:

a) The work that Intel has done certainly can benefit more then just Office. Appload time is an issue for the PC platform at large and by putting it in the OS this benefit is accrued broadly. If this capability is designed specifically around Office then the end-user will probably not benefit from the rest of their apps on their system. This need not be the case. The work that Intel has done is good enough so that the "optimization" does not degrade significantly across a set of applications (as opposed to a prioritized list). If Office implements the tool as a system wide benefit, call me a traditionalist but it just seems like an odd end-user experience. Install Office, then get prompted to walk thru a series of steps to identify the applications that you run (w/ Office apps at the top of the list of course) and then defrag your disk.
b) Attached below is a write-up from BillKru, who is leading our technical evaluation of Intel's work, on what Intel has done and how we would refine into a system offering. To achieve the best appload time optimization means a new app defragger which we are doing now, FAT(32), and changes to the way that Intel tool logs access to the disk. The Memphis team is committed to doing these refinements to these system tools in conjunction w/ the work from Intel. The end-user experience is along the lines of a system "tune-up" utility that sometime after upgrade recommends to the user a series of things to improve the performance & efficiency of their system. The most critical element is disk optimization. Subsequently, on a periodic basis, we will profile the users app usage over an extended period of time to ensure our optimizations are consistent with what the user is currently doing (eg. they may have installed an app).

c) Intel's distribution capabilities: Intel of course is very anxious to get credit for their work. If they licensed w/ Office, we would have a harder time preventing them from licensing to everyone under the sun. This has two limitations; positioning Intel as a great sw company, mitigating the amount of "benefit" accrued directly to MS.

-----Original Message-----
From: Marshall Brumer
Sent: Friday, May 02, 1997 2:17 PM
To: Bill Gates
Cc: Aaron Contorer; Bill Veghte; Carl Stork (Exchange); Mike Porter
Subject: RE: Applications boot time

We have completed an LOI to put the application launch technology for disks into Memphis. We have not completed an agreement that allows us to have the code here. The agreement is in process and should be complete soon.

The office folks can already start testing with the MS version of this code in Memphis. This will be the base that we use to apply the Intel technology to.

You need to decide if this becomes a feature for Memphis or something that the apps folks ship as part of their setup - billv can add here.

-----Original Message-----
From: Bill Gates
Sent: Thursday, May 01, 1997 7:20 PM
To: Marshall Brumer
Cc: Aaron Contorer
Subject: FW: Applications boot time

Do we have the Intel code yet? Did we reach a good agreement on this?

Every day that goes by without this is bad. We need to get going.

-----Original Message-----
From: Wael Bahaa-El-Din
Sent: Thursday, May 01, 1997 12:42 PM
To: Bob Fitzgerald; Jon DeVaan
Cc: David Fields (NT); Stephen Hsiao; Bill Gates; Jim Allchin (Exchange); Moshe Dunie; David Cutler; Lou Perazzoli; Mark Lucovsky; Rick Rashid; Duane Campbell; Antoine Leblond; Mark Walker (Word); Jim Walsh
Subject: RE: Applications boot time

Jon and Fitz,

1. We can measure the gain with what Fitz has already (a list of hints to the loader to touch pages in a certain order). However, Fitz's method may read pages that doesn't get referenced (as he mentioned), this will not yield good performance for 16MB which is very sensitive to %waste. If we get a properly BBTized build for Word using the startup scenario, we can do the best possible experiment for the app set of images (minimize the pages read, read in big 32K or 64K). We can do the experiment as soon as the office group provides us with the proper BBTized image.

2. Startup is by far the most important operation impacting "perceived performance" We need to BBTize using startup (in addition to open/save/print) with improved weights in the scenarios and making sure that page placement is satisfactory by viewing the outcome with our tools. We would like to depend on your group and the BBT group to help us. We will initiate a meeting next week with our group, Jim Walsh, and Fitz.

3. Anyone thinking about reducing the number of pages that get referenced? Yes!

a. We are BBTizing the NT5.0 kernel for the first time with a lot of help from Fitz and Hoi from the BBT group. This should reduce the code pages referenced dramatically in the paged sections of the Kernel.

b. We are working on reducing the pages touched from the registry, page tables, non-paged pool, font files, system threads, etc.

c. David Fields & NTW perf team will continue working with the Office people to give them feedback in optimizing for NT as they did with Mark Walker for Word 8.0 to reduce the boot from 13 to 6 seconds in 16MB.

4. In 16 Mbyte memory, we measure an average of 8KB/read (Fitz's number of 20 KB/read is measured in 32 MB memory where NT uses a larger cluster).

Thanks,

Wael

-----Original Message-----
From: Bob Fitzgerald
Sent: Wednesday, April 30, 1997 2:05 PM
To: Wael Bahaa-El-Din; Jon DeVaan
Cc: David Fields (NT); Stephen Hsiao; Bill Gates; Jim Allchin (Exchange); Moshe Dunie; David Cutler; Lou Perazzoli; Mark Lucovsky; Rick Rashid; Duane Campbell; Antoine Leblond; Mark Walker (Word); Jim Walsh
Subject: RE: Applications boot time

The reference scattering in winword (probably) means that code was getting executed during boot in the WinNT perf lab that wasn't getting executed at boot in the Office build lab. The existing lego training process (scenarios, platforms) may not adequately prepare the optimized binaries for the variety of target platforms they will ultimately run on. Some of this may be fixable.

We should consider the possibility that some fragmentation -- skew between training scenarios and real customers -- is unavoidable and that we need to tolerate some modest amount of fragmentation and still boot quickly.

I've done some approximations of gang loading -- reading all the necessary pages of winword.exe in address order, then all the pages of mso97.dll, etc. The key lessons are:

* large block reads (64 KB) maximize disk bandwidth. The ~5 page per read average with the WinNT 4.0 cluster pager (cluster size 7-8) delivers as little as 1/2 the disk bandwidth available with 16 page reads. (this depends a lot on the particular disk you measure)
* no overruns, as any unnecessary pages read increase memory pressure and increase paging. We currently overrun a lot.
* locality matters, little clumps of one or two pages drag down the average disk read size and bandwidth and increase overruns
* reading in address order instead of fault order increases the apparent locality, increases read size and decreases overruns

It may not be necessary to create a special "boot section" to make gang loading work -- just knowing the list of pages needed is enough. The list could live anywhere, e.g. in the registry as the Win97 folks may be considering. To prevent overruns, it is important to know when to stop, i.e. to keep the WinNT cluster pager from reading off the end of each island of contiguous page references. Clever disk layout may help to increase disk read sizes or to reduce seek delays.

Coalescing the dozen or so system .DLLs (kernel32.dll, user32.dll, gdi32.dll, shell32.dll, ole32.dll, ...) that are used into a single .DLL would decrease footprint and increase locality.

From: Jon DeVaan
Sent: Wednesday, April 30, 1997 10:25

AM

To: Wael Bahaa-El-Din
Cc: David Fields (NT); Stephen Hsiao; Bill Gates; Jim Allchin (Exchange); Moshe Dunie; David Cutler; Lou Perazzoli; Mark Lucovsky; Rick Rashid; Bob Fitzgerald; Duane Campbell; Antoine Leblond; Mark Walker (Word); Jim Walsh
Subject: RE: Applications boot time

I'm surprised by the fragmentation of the accesses in winword.exe given that we already bbt word. My first guess is that compromises to boot are introduced as we add usage scenarios to the builds. It's an area definitely worth looking into.

Is anyone thinking about reducing the number of pages that get referenced? We did work in Office 97 to eliminate system calls made, but it looks like there ought to be many more to eliminate. I don't think we can do all of the work here though. MarkL had a list of redundant api calls in Office 95 that we made good progress in fixing in Office 97. That was very helpful. If you guys had a list of, "Why are you calling this api?" for apis that are suspicious to you, we could put it to good use.

I wonder if we can do a good simulation of the effect of boot block before we get the other things done. One experiment that ought to not be too hard to do, is to hack loadmodule to load the boot section. For the experiment just estimate a block using the existing exe. Word has a pretty reasonable set of 3 or 4 blocks at the beginning of the exe that could be called the boot sector. Also for the experiment, just use the module name to see if it is winword.exe, or mso97.dll and their associated intl dlls, then set the read ahead amount to the boot block size, touch the first page, set the read ahead size back, and touch all the other pages in the boot block. (Maybe there is a better way to make the read ahead pages not get discarded right away.) Seeing the effect of this would be very interesting. Of course my description of how to do this is na?ve. If you see a better way, then great.

-----Original Message-----
From: Wael Bahaa-El-Din
Sent: Tuesday, April 29, 1997 8:13 PM
To: Bill Gates; Jim Allchin (Exchange); Moshe Dunie; Jon DeVaan; David Cutler; Lou Perazzoli; Mark Lucovsky; Rick Rashid; Bob Fitzgerald
Cc: David Fields (NT); Stephen Hsiao
Subject: FW: Applications boot time
Importance: High

This is a follow-up on BillG request to optimize app startup:

In the report enclosed below, we discuss both Word8 startup running on NT4.0 and how we can improve it. The startup takes 6.04 seconds on 100Mhz Pentium with 16 MBytes of memory because it involves reading 3 Mbytes from around 50 files on disk with a poor IO rate of 631 KB/s.We would like to be able to perform the startup by reading 2Mbytes (pages that are really referenced) with an IO rate of 2Mbytes/s. The startup also involves CPU consumption of 1.5 seconds. We show that the result of the current BBTizing (due to the scenarios used) can be significantly improved. With the improved BBTizing and a change to use larger clusters during application startup, we should be able to get a much higher bandwidth for disk reads (around 1.5-2MB/s) for an overall boot time of 3.0 seconds in 16 Mbytes. After working on the proper BBTizing of WORD image (and other office applications), we will investigate other solutions to improve the I/O bandwidth such as reorganizing the on-disk locality of the files and constructing a boot section for the application code/data.

Thanks,

Wael

P.S. Last figure shows the current impact of BBT on WORD.exe. It doesn't print on some printers but you can see you on your monitor.

<< File: word8bootupdate.doc >>

-----Original Message-----
From: Moshe Dunie
Sent: Tuesday, March 18, 1997 9:25 AM
To: Lou Perazzoli; Wael Bahaa-El-Din
Subject: FW: Applications boot time
Importance: High

Can you please do this experiment with the office folks.

Thanks.....Moshe

-----Original Message-----
From: Bill Gates
Sent: Tuesday, March 18, 1997 8:54 AM
To: Moshe Dunie
Cc: Jon DeVaan; Jim Allchin (Exchange)
Subject: FW: Applications boot time

Can you have someone from NT work with the Office group on point #1 here?

I think Office boot times are critical to our future and I am pushing for more innovation in this area.

-----Original Message-----
From: Jon DeVaan
Sent: Monday, March 17, 1997 10:13 PM
To: Bill Gates
Cc: Richard Fade; Steven Sinofsky; Brad Silverberg; Nathan Myhrvold; Aaron Contorer; Rick Rashid
Subject: RE: Applications boot time

Two things would be extremely helpful for making this come true.

1) I am embarrassed to report that we still do not have agreement from the OS teams to declare a boot section in an exe and load it all at once. This would be a major improvement. (OK, perhaps a wild assertion on my part) The argument against this is usually along the lines of "we tried writing a tight loop that paged in x bytes of code in the app and it didn't help boot time any." My argument against this is, that experiment does not cause x-bytes to happen with exactly one IO operation. I want the NT guys to run this experiment: Change the gang-load size parameter to be the boot size of an exe for the first fault, then change it back dynamically. This is the right experiment to run. I can't convince anyone to do this experiment.

2) We need Lego v. 2. Lego has been a big help, but it has a bunch of inadequacies. I was surprised to learn that it cannot do code groupings based on scenario. What I mean is, I want to know for n operations (boot, file open, file save, file print) the set of basic blocks used in each operation. Then I want the code in my exe distributed so that the code that is boot only is one contiguous block, boot AND file open in one contiguous block next to that, the code that is boot AND file save next to that, etc..., all n! blocks defined. Then I want those blocks ordered so that by priority of operation I have one contiguous block of code for the highest priority

operations and then 2, 3, 4, or more blocks for operations as priority wanes. Lego can't do this today.

It is also fair to note 2nd boot of an app on win95 or NT are typically 4-5x faster than first boot. (i.e. 80% of boot time is page faulting)

-----Original Message-----
From: Bill Gates
Sent: Sunday, March 16, 1997 10:20 AM
To: Jon DeVaan
Cc: Richard Fade; Steven Sinofsky; Brad Silverberg; Nathan Myhrvold; Aaron Contorer
Subject: FW: Applications boot time

One goal I think has to be totally crucial for Office 9X is to get boot times well below 10 seconds.

I know this will require invention and work with the OS and even rethinking how we use DLLs but I think it's a requirement.

Office feels heavy for a number of reasons but the one that you really notice is the applications boot time.

-----Original Message-----
From: Rick Rashid
Sent: Saturday, March 15, 1997 12:48 PM
To: Bill Gates; Aaron Contorer; Darryl Rubin
Cc: Jim Allchin (Exchange); Steven Sinofsky; Butler Lampson; Nathan Myhrvold
Subject: RE: Applications boot time

I'll look into this again, but it was my impression that with the last round of LEGO work which already allows Office to linearize its initial page faults and with way NT handles paging that we were already getting about all we could get in terms of loading speed. Assuming fairly linear accesses, a disk should be just as good as a 100MB ethernet and probably better.

The biggest loading issue, I suspect, is related to the fact that the "access set" of a Windows NT system with Office is much larger than 24MB (actually its larger I believe that 32MB) and that there is going to be paging going on other than just paging in the application.

Also, I believe, there is considerable CPU time (seconds) devoted to "startup" in the apps as they open files, review registry entries, link things, allocate space, etc. Nothing done to data load times will help make this go away.

I've also noticed that there is also a lot of "hidden" access to servers and devices which typically justs times out. When I run Office on the machine I have which has a zip drive, for example, it routinely spins up the drive for no obvious reason. Likewise I will often hear a random floppy access or see the system pause when I'm not connected to a network.

From: Darryl Rubin
Sent: Friday, March 14, 1997 10:18 PM
To: Bill Gates; Aaron Contorer
Cc: Jim Allchin (Exchange); Steven Sinofsky; Butler Lampson; Nathan Myhrvold; Rick Rashid
Subject: RE: Applications boot

time

There are also ways to improve the illusion of startup speed. It should be very easy for an app to put up what looks like the app window and the first page of the document (or the page the user last visited), even if this is mostly a smoke-and-mirrors show until more of the app loads to make it live. The app could also be restructured to prioritize which parts of the UI come alive first, based on what operations are the first that users usually try (scroll? Pull down file or edit menu?). Of course we should be making the initial working set of the app smaller and also do caching tricks like you suggest. I think that plus tricks could result in a dramatic improvement in perceived boot performance.

-----Original Message-----
From: Bill Gates
Sent: Friday, March 14, 1997 9:35 PM
To: Aaron Contorer
Cc: Jim Allchin (Exchange); Steven Sinofsky; Darryl Rubin; Butler Lampson; Nathan Myhrvold; Rick Rashid
Subject: Applications boot time

I am hard core about trying to find ways to make our applications boot faster. We have to do it. It's the whole reason people think our applications are too big.

The question I have is what if the server had say the most commonly used 24 megabytes of Office in Ram in a form that made it very easy to get to. Would it be faster over a 100megabit fairly unloaded Ethernet to get these bits across the network? The idea is basically the Berkeley NOW approach except without the low latency network which makes it such a big win for them. I wonder what tricks might allow this to work well. Reducing latency is a worthy project for many reasons.

<< File: intel exec report.doc >>