From: Steven Sinofsky
Sent: Wednesday, January 27, 1999 5:13 PM
To: Rick Rashid
Subject: RE: Applications boot time (1 s Word 1st-boot)

We were going to do this for Office 95 by just reading the EXE and having a large file cache -- this forced it to be in the file I/O ram cache. The problem was that we kicked out important things and it was hard to know which EXEs to really page in if the system had any RAM contraints.

-----Original Message-----
From: Rick Rashid
Sent: Wednesday, January 27, 1999 9:48 AM
To: Bill Gates; Amitabh Srivastava; Arthur Zwiegincew
Cc: Eric Rudder; Jon DeVaan; Steven Sinofsky; Nathan Myhrvold; Mark Lucovsky; Jim Walsh
Subject: RE: Applications boot time (1 s Word 1st-boot)

On systems with enough RAM under Win98 you can allocate a ramdisk and put the core office files into it and get the same effect described below - nearly instant start times. I tried this on one of my machines with 128MB of RAM. I partitioned the ram into 64MB system and 64MB ramdisk. The ramdisk I used was a freeware DOS 6.x XMS tsr that I pulled off the Internet. I load the ramdisk with the relevant files during bootup in my autoexec.bat file from a temp directory and set my paths so that the ramdisk is the first one on the list. Works like a charm (although getting the right files together isn't trivial). Generally speaking, running Excel, Powerpoint and Word becomes a < 1sec task - more like what you get when you have the files "cached" from a previous run - largely independent of previous activity such as running Visual Studio builds.

Of course, you could optimize this quite a bit. You seem to need about 30MB for the "largest" of the exes and dlls (e.g. Winword.exe, excel.exe, mso97.dll). You could imagine a ramdisk that uses compression (we may already have one lying around but I couldn't find it in my 5 minute search of the Internet) and get a factor of two compression so you could probably get away with less than 15MB of actual RAM. A specialized "office" compressor could probably get a factor of 3. You could also "preload" the cache from a contiguous part of disk rather than getting it from normal DOS file copies.

I mentioned this to Jim Walsh. He said they did some experiments with ram caches a while back but he is going to check into it again. Of course, this isn't the most efficient way to use memory and an idea like this won't help a low memory system. The main thing that has changed is the increasing prevalence of large memory systems due to low memory prices.

Bob Fitzgerald did something along these lines for the NT team as an experiment to help them speed up boot. He built a RAM boot cache which preloads the physical disk blocks needed by NT during the idle times when NT is waiting for device probes to complete. Although they don't plan to ship it for schedule reasons, the NT team has used this code profitably to help find non-disk bottlenecks in their boot procedures.

-Rick

-----Original Message-----
From: Bill Gates
Sent: Sunday, January 24, 1999 5:48 PM
To: Amitabh Srivastava; Arthur Zwiegincew
Cc: Eric Rudder; Jon DeVaan; Steven Sinofsky; Nathan Myhrvold; Mark Lucovsky; Rick Rashid
Subject: FW: Applications boot time (1 s Word 1st-boot)

The value of applications being able to boot at this kind of speed is totally HUGE HUGE HUGE. This alone would get most users to upgrade. I want to totally understand ho we make this happen.

I do NOT understand why we would need any new hooks in NT to do log the file of pages we need. Vulcan should be able to gather all the information that Andrew thinks we need - the page fault list during the boot up time.

When you do an install of Word if you are not starved for disk space we would run a piece of boot that would set up the "boot file" and make sure to enable the code that uses the boot file.

There may need to be something in NT to have the boot file read and the memory map be set up the right way. Has anyone figure out what work would be required for this?
I think its worth doing so ASAP.

Am I missing something here where we really need to change NT more than just a special load?

Whenever I get mail like this I am reminded of how SUPRISED I am that no one ever suggests what we should do about the registry - where do we go to have something that is less of a problem in terms of management, deployment and speed. It blows the mind.

Getting app start up speed to be this fast would be super fantastic. Lets figure out how to make this real!

-----Original Message-----
From: Nathan Myhrvold
Sent: Sunday, January 24, 1999 4:18 PM
To: Bill Gates
Subject: FW: Applications boot time (1 s Word 1st-boot)

FYI - doing things like splitting DLLs then remerging another way, cloning small sections if need be is a great example of being able to manipulate a large source base more effectively with tools that with people.

Nathan

-----Original Message-----
From: Amitabh Srivastava
Sent: Sunday, January 24, 1999 9:28 AM
To: Nathan Myhrvold
Cc: Rick Rashid
Subject: RE: Applications boot time (1 s Word 1st-boot)

For Office, Arthur is right the behavior for Office is that we keep bouncing back and forth between ms09 and winword. There is little you can do. Our opinion is Office has a reverse situation from NT we need dll-splitting followed by dll-merging : MS09 is too big. We need to analyze MS09 and break it into few parts. The part which is specific for Word should be dll-merged with Word, the part specific to Excel should be dll-merged with Excel and so on. The part that is shared can be kept as a separate dll. It is also possible to clone critical pieces. Our hope is that with this approach we can stop the bouncing back and forth. We will do an analysis and determine what the structure should be.

We are trying to get a new version of BBT and Vulcan released to the company at the end of this month. NT has been able to optimize 50 more dlls over what they ever did before. Following this, We plan to study dll-merging and make if very effective. We'll be studying Office, SQL .. more closely – our focus lately has been on NT. Regardless, all this will work with Arthur's scheme.

Amitabh

-----Original Message-----
From: Nathan Myhrvold
Sent: Friday, January 22, 1999 11:15 PM
To: Amitabh Srivastava
Cc: Rick Rashid
Subject: FW: Applications boot time (1 s Word 1st-boot)

-----Original Message-----
From: Bill Gates
Sent: Friday, January 22, 1999 1:39 PM
To: Eric Rudder; Nathan Myhrvold; Rick Rashid
Subject: FW: Applications boot time (1 s Word 1st-boot)

FYI....

-----Original Message-----
From: Arthur Zwiegincew
Sent: Thursday, January 21, 1999 6:50 PM
To: Bill Gates; Jon DeVaan; Jim Allchin (Exchange)
Subject: RE: Applications boot time (1 s Word 1st-boot)

Re your wishes below: I'm from Office Perf and I've come up with an idea to radically speed up app launch and other scenarios. It's currently being patented (MS#116278.1). For it to succeed, we need NT's blessing (it requires some changes in VMM). I have

solutions that would work with (1) Office/MS apps only, (2) any random app.

Bottom line:
Word9 launch on a clean system (i.e. registry in memory) with enough RAM (maybe 32MB) and a very slow disk (3MB/s), should take 0.32 s (streaming+decompression; see below) + CPU time (less than 0.5 s) + random I/O time + Explorer time ~= 1-1.5 s. Radical.

In short:
The idea is to instrument the page fault handler. Apps call MmBeginScenario(GUID, ...) and MmEndScenario(GUID) to tell VMM that they're executing common scenarios (optionally, in the future we could log all pf's and run idle pattern-matching to find scenarios w/o apps' involvement). VMM logs pf's as they occur, and then, at idle, all pages that were faulted are copied into a scenario file. The next time this scenario is run, VMM reads everything in one I/O and sets up PTEs/VADs for all pages in the scenario (we'll use NTFS defrag hooks to make sure this really is one disk I/O). When the pages are actually needed, they'll be soft-faulted into the app working set. This is the holy grail--we're reading exactly the set of EXE/DLL (incl. system DLLs) pages that we need. All remaining I/O is random stuff, like normal.dot, and the registry (which totally sucks, they need some big-time, major fixes to the registry; on a loaded system there are nearly as many registry I/Os as there are winword.exe I/Os!!! I have some ideas how to help Office re registry). There are a bunch of issues, but the ones I have thought about, I was able to resolve.

All other approaches I know of, either complement this (e.g. BBT), or are inferior (e.g. Tune-Up Wizard).

An early spec is here:
<< File: spec.doc >>
Please try to convince NT to spare some resources. If they can't, I'd be happy to write this stuff myself, but I would have to have DAD's blessing (like I said, I'm from Office).

Re stuff you mention below:
* Gang-loading from user mode (JonDe point 1) does work at disk-speed, even though we're issuing zillions of IRPs. As long as they come with a high-enough frequency, we can keep the disk spinning, and achieve awesome throughput (almost identical to pure streaming). Problem: we're not even close to touching 100% of boot-only BBT, so (1) gang-loading takes more time than it should, (2) there's memory pressure and we hit the pagefile a lot. Bottom line: no gains. I ran my tests on a P5-100, 32MB, NT4, shitty SCSI disk, recent winword9 build. If you're interested, I have a bunch of charts and data on this (I spent a good portion of my life on this!). BTW, it would be nice if we went open-source within MS (or at least DAD and NT)-idea recommended by VinodV. It would have been much easier if I had NT sources at the time.
* Contiguous BBT sections (JonDe point 2) won't help much by themselves, but would work perfectly with my prefetcher. The problem is that reads from winword.exe are interrupted by reads from mso9.dll, system DLLs, reg accesses, etc. The bottom line is that avg I/O time during winword launch is almost = disk seek time! Unbelievable, but true.
* Putting up a mock-up of the app window (DarrylR's point) is a good idea, but usability tests have shown that users do not like this (or so they think; when focus group studies were conducted with the first Ford Taurus in the early '80s, users said they didn't like its new looks, but when Ford started selling them, people changed their minds; go figure). There were other proposals in Office, such as a rich splash screen, actually a welcome screen, with a progress bar and options: new file, new from template, open file (+ listbox), etc. Users rejected it too. You can check it out on http://office10/bin/tables.asp?docType=Prototype&sortBy=Date (check out my prototype while you're there--it's the 1st one on the list).
* Winword9 launch times on Hydra are amazing (but obvious): less than 2 s on wtsoff. Hydra offers tons of completely new possibilities for mega-heavy optimizations.

Pls let me know what you think. thx

Arthur Zwiegincew
** Hardcore Computer Maniac **

-----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.

From: Steven Sinofsky
Sent: Sunday, January 24, 1999 5:57 PM
To: Bill Gates
Cc: Jon DeVaan
Subject: RE: Applications boot time (1 s Word 1st-boot)

Sorry he sent this to you. He is a new hire who is incredibly enthusiastic. It meant the world that you answered him.

We don't need hooks in NT. There are some things we should get in kernel/loader that we had in Win3.x that we continue to talk about (gang loading). I believe there might be something like this in Windows 2000.

We know boot is a huge win. We just spend a huge amount of time not getting worse. We also chased the disk defragger solution a lot this release (though we do ship an office-specific version of it for Win95 in our box).

This continues to be an incredibly hard to solve problem. Boot is becoming much more data-driven than code-driven. There are so many things that are conditional based on the document, the configuration, etc. The registry is just one central place where we go for all those things. Even something as simple as anti-piracy ends up impacting boot. And of course all the resiliency work had a minor impact.

Miraculously, our boot time is essentially the same for Office 2000 as it was for Office 97. But I think when you use the product you will notice some subtle things that we did to help us boot faster, but might not make for a better overall experience. We delay load any DLLs and features you might not need. This means after you boot at the first idle time we continue the process of loading some of the code -- the speller in word is a good example, or the office assistant. This means that most of the time you see a faster boot, but if you boot and then don't do anything for a few seconds the background load kicks in. This might slow you down if you want to start typing.

At our planning retreat last week, many people brought up the idea of backing off the registry completely. It kills us on so many things. But the downside is that the key to Windows 2000's managability and especially lockdown is all in the registry. Of cours there are all the legacy issues of OLE and COM that require the registry as well.

This is an area we will do better on for Office10.

-----Original Message-----
From: Bill Gates
Sent: Sunday, January 24, 1999 5:48 PM
To: Amitabh Srivastava; Arthur Zwiegincew
Cc: Eric Rudder; Jon DeVaan; Steven Sinofsky; Nathan Myhrvold; Mark Lucovsky; Rick Rashid
Subject: FW: Applications boot time (1 s Word 1st-boot)

The value of applications being able to boot at this kind of speed is totally HUGE HUGE HUGE. This alone would get most users to upgrade. I want to totally understand ho we make this happen.

I do NOT understand why we would need any new hooks in NT to do log the file of pages we need. Vulcan should be able to gather all the information that Andrew thinks we need - the page fault list during the boot up time.

When you do an install of Word if you are not starved for disk space we would run a piece of boot that would set up the "boot file" and make sure to enable the code that uses the boot file.

There may need to be something in NT to have the boot file read and the memory map be set up the right way. Has anyone figure out what work would be required for this?
I think its worth doing so ASAP.

Am I missing something here where we really need to change NT more than just a special load?

Whenever I get mail like this I am reminded of how SUPRISED I am that no one ever suggests what we should do about the registry - where do we go to have something that is less of a problem in terms of management, deployment and speed. It blows the mind.

Getting app start up speed to be this fast would be super fantastic. Lets figure out how to make this real!

-----Original Message-----
From: Nathan Myhrvold

Sent: Sunday, January 24, 1999 4:18 PM
To: Bill Gates
Subject: FW: Applications boot time (1 s Word 1st-boot)

FYI - doing things like splitting DLLs then remerging another way, cloning small sections if need be is a great example of being able to manipulate a large source base more effectively with tools that with people.

Nathan

-----Original Message-----
From: Amitabh Srivastava
Sent: Sunday, January 24, 1999 9:28 AM
To: Nathan Myhrvold
Cc: Rick Rashid
Subject: RE: Applications boot time (1 s Word 1st-boot)

For Office, Arthur is right the behavior for Office is that we keep bouncing back and forth between ms09 and winword. There is little you can do. Our opinion is Office has a reverse situation from NT we need dll-splitting followed by dll-merging : MS09 is too big. We need to analyze MS09 and break it into few parts. The part which is specific for Word should be dll-merged with Word, the part specific to Excel should be dll-merged with Excel and so on. The part that is shared can be kept as a separate dll. It is also possible to clone critical pieces. Our hope is that with this approach we can stop the bouncing back and forth. We will do an analysis and determine what the structure should be.

We are trying to get a new version of BBT and Vulcan released to the company at the end of this month. NT has been able to optimize 50 more dlls over what they ever did before. Following this, We plan to study dll-merging and make if very effective. We'll be studying Office, SQL .. more closely -- our focus lately has been on NT. Regardless, all this will work with Arthur's scheme.

Amitabh

-----Original Message-----
From: Nathan Myhrvold
Sent: Friday, January 22, 1999 11:15 PM
To: Amitabh Srivastava
Cc: Rick Rashid
Subject: FW: Applications boot time (1 s Word 1st-boot)

-----Original Message-----
From: Bill Gates
Sent: Friday, January 22, 1999 1:39 PM
To: Eric Rudder; Nathan Myhrvold; Rick Rashid
Subject: FW: Applications boot time (1 s Word 1st-boot)

FYI....

-----Original Message-----
From: Arthur Zwiegincew
Sent: Thursday, January 21, 1999 6:50 PM
To: Bill Gates; Jon DeVaan; Jim Allchin (Exchange)
Subject: RE: Applications boot time (1 s Word 1st-boot)

Re your wishes below: I'm from Office Perf and I've come up with an idea to radically speed up app launch and other scenarios. It's currently being patented (MS#116278.1). For it to succeed, we need NT's blessing (it requires some changes in VMM). I have solutions that would work with (1) Office/MS apps only, (2) any random app.

Bottom line:
Word9 launch on a clean system (i.e. registry in memory) with enough RAM (maybe 32MB) and a very slow disk (3MB/s), should take 0.32 s (streaming+decompression; see below) + CPU time (less than 0.5 s) + random I/O time + Explorer time ~= 1-1.5 s. Radical.

In short:
The idea is to instrument the page fault handler. Apps call MmBeginScenario(GUID, ...) and MmEndScenario(GUID) to tell VMM that they're executing common scenarios (optionally, in the future we could log all pf's and run idle pattern-matching to find

scenarios w/o apps' involvement). VMM logs pf's as they occur, and then, at idle, all pages that were faulted are copied into a scenario file. The next time this scenario is run, VMM reads everything in one I/O and sets up PTEs/VADs for all pages in the scenario (we'll use NTFS defrag hooks to make sure this really is one disk I/O). When the pages are actually needed, they'll be soft-faulted into the app working set. This is the holy grail--we're reading exactly the set of EXE/DLL (incl. system DLLs) pages that we need. All remaining I/O is random stuff, like normal.dot, and the registry (which totally sucks, they need some big-time, major fixes to the registry; on a loaded system there are nearly as many registry I/Os as there are winword.exe I/Os!!! I have some ideas how to help Office re registry). There are a bunch of issues, but the ones I have thought about, I was able to resolve.

All other approaches I know of, either complement this (e.g. BBT), or are inferior (e.g. Tune-Up Wizard).

An early spec is here:
<< File: spec.doc >>
Please try to convince NT to spare some resources. If they can't, I'd be happy to write this stuff myself, but I would have to have DAD's blessing (like I said, I'm from Office).

Re stuff you mention below:
* Gang-loading from user mode (JonDe point 1) does work at disk-speed, even though we're issuing zillions of IRPs. As long as they come with a high-enough frequency, we can keep the disk spinning, and achieve awesome throughput (almost identical to pure streaming). Problem: we're not even close to touching 100% of boot-only BBT, so (1) gang-loading takes more time than it should, (2) there's memory pressure and we hit the pagefile a lot. Bottom line: no gains. I ran my tests on a P5-100, 32MB, NT4, shitty SCSI disk, recent winword9 build. If you're interested, I have a bunch of charts and data on this (I spent a good portion of my life on this!). BTW, it would be nice if we went open-source within MS (or at least DAD and NT)-idea recommended by VinodV. It would have been much easier if I had NT sources at the time.
* Contiguous BBT sections (JonDe point 2) won't help much by themselves, but would work perfectly with my prefetcher. The problem is that reads from winword.exe are interrupted by reads from mso9.dll, system DLLs, reg accesses, etc. The bottom line is that avg I/O time during winword launch is almost = disk seek time! Unbelievable, but true.
* Putting up a mock-up of the app window (DarrylR's point) is a good idea, but usability tests have shown that users do not like this (or so they think; when focus group studies were conducted with the first Ford Taurus in the early '80s, users said they didn't like its new looks, but when Ford started selling them, people changed their minds; go figure). There were other proposals in Office, such as a rich splash screen, actually a welcome screen, with a progress bar and options: new file, new from template, open file (+ listbox), etc. Users rejected it too. You can check it out on http://office10/bin/tables.asp?docType=Prototype&sortBy=Date (check out my prototype while you're there--it's the 1st one on the list).
* Winword9 launch times on Hydra are amazing (but obvious): less than 2 s on wtsoff. Hydra offers tons of completely new possibilities for mega-heavy optimizations.

Pls let me know what you think. thx

Arthur Zwiegincew
** Hardcore Computer Maniac **

-----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.

From: Bill Gates
Sent: Sunday, January 24, 1999 5:48 PM
To: Amitabh Srivastava; Arthur Zwiegincew
Cc: Eric Rudder; Jon DeVaan; Steven Sinofsky; Nathan Myhrvold; Mark Lucovsky; Rick Rashid
Subject: FW: Applications boot time (1 s Word 1st-boot)

The value of applications being able to boot at this kind of speed is totally HUGE HUGE HUGE. This alone would get most users to upgrade. I want to totally understand ho we make this happen.

I do NOT understand why we would need any new hooks in NT to do log the file of pages we need. Vulcan should be able to gather all the information that Andrew thinks we need - the page fault list during the boot up time.

When you do an install of Word if you are not starved for disk space we would run a piece of boot that would set up the "boot file" and make sure to enable the code that uses the boot file.

There may need to be something in NT to have the boot file read and the memory map be set up the right way. Has anyone figure out what work would be required for this?
I think its worth doing so ASAP.

Am I missing something here where we really need to change NT more than just a special load?

Whenever I get mail like this I am reminded of how SUPRISED I am that no one ever suggests what we should do about the registry - where do we go to have something that is less of a problem in terms of management, deployment and speed. It blows the mind.

Getting app start up speed to be this fast would be super fantastic. Lets figure out how to make this real!

-----Original Message-----
From: Nathan Myhrvold
Sent: Sunday, January 24, 1999 4:18 PM
To: Bill Gates
Subject: FW: Applications boot time (1 s Word 1st-boot)

FYI - doing things like splitting DLLs then remerging another way, cloning small sections if need be is a great example of being able to manipulate a large source base more effectively with tools that with people.

Nathan

-----Original Message-----
From: Amitabh Srivastava
Sent: Sunday, January 24, 1999 9:28 AM
To: Nathan Myhrvold
Cc: Rick Rashid
Subject: RE: Applications boot time (1 s Word 1st-boot)

For Office, Arthur is right the behavior for Office is that we keep bouncing back and forth between ms09 and winword. There is little you can do. Our opinion is Office has a reverse situation from NT we need dll-splitting followed by dll-merging : MS09 is too big. We need to analyze MS09 and break it into few parts. The part which is specific for Word should be dll-merged with Word, the part specific to Excel should be dll-merged with Excel and so on. The part that is shared can be kept as a separate dll. It is also possible to clone critical pieces. Our hope is that with this approach we can stop the bouncing back and forth. We will do an analysis and determine what the structure should be.

We are trying to get a new version of BBT and Vulcan released to the company at the end of this month. NT has been able to optimize 50 more dlls over what they ever did before. Following this, We plan to study dll-merging and make if very effective. We'll be studying Office, SQL .. more closely – our focus lately has been on NT. Regardless, all this will work with Arthur's scheme.

Amitabh

-----Original Message-----
From: Nathan Myhrvold
Sent: Friday, January 22, 1999 11:15 PM

To: Amitabh Srivastava
Cc: Rick Rashid
Subject: FW: Applications boot time (1 s Word 1st-boot)

-----Original Message-----
From: Bill Gates
Sent: Friday, January 22, 1999 1:39 PM
To: Eric Rudder; Nathan Myhrvold; Rick Rashid
Subject: FW: Applications boot time (1 s Word 1st-boot)

FYI....

-----Original Message-----
From: Arthur Zwiegincew
Sent: Thursday, January 21, 1999 6:50 PM
To: Bill Gates; Jon DeVaan; Jim Allchin (Exchange)
Subject: RE: Applications boot time (1 s Word 1st-boot)

Re your wishes below: I'm from Office Perf and I've come up with an idea to radically speed up app launch and other scenarios. It's currently being patented (MS#116278.1). For it to succeed, we need NT's blessing (it requires some changes in VMM). I have solutions that would work with (1) Office/MS apps only, (2) any random app.

Bottom line:
Word9 launch on a clean system (i.e. registry in memory) with enough RAM (maybe 32MB) and a very slow disk (3MB/s), should take 0.32 s (streaming+decompression; see below) + CPU time (less than 0.5 s) + random I/O time + Explorer time ~= 1-1.5 s. Radical.

In short:
The idea is to instrument the page fault handler. Apps call MmBeginScenario(GUID, ...) and MmEndScenario(GUID) to tell VMM that they're executing common scenarios (optionally, in the future we could log all pf's and run idle pattern-matching to find scenarios w/o apps' involvement). VMM logs pf's as they occur, and then, at idle, all pages that were faulted are copied into a scenario file. The next time this scenario is run, VMM reads everything in one I/O and sets up PTEs/VADs for all pages in the scenario (we'll use NTFS defrag hooks to make sure this really is one disk I/O). When the pages are actually needed, they'll be soft-faulted into the app working set. This is the holy grail--we're reading exactly the set of EXE/DLL (incl. system DLLs) pages that we need. All remaining I/O is random stuff, like normal.dot, and the registry (which totally sucks, they need some big-time, major fixes to the registry; on a loaded system there are nearly as many registry I/Os as there are winword.exe I/Os!!! I have some ideas how to help Office re registry). There are a bunch of issues, but the ones I have thought about, I was able to resolve.

All other approaches I know of, either complement this (e.g. BBT), or are inferior (e.g. Tune-Up Wizard).

An early spec is here:
<< File: spec.doc >>
Please try to convince NT to spare some resources. If they can't, I'd be happy to write this stuff myself, but I would have to have DAD's blessing (like I said, I'm from Office).

Re stuff you mention below:
* Gang-loading from user mode (JonDe point 1) does work at disk-speed, even though we're issuing zillions of IRPs. As long as they come with a high-enough frequency, we can keep the disk spinning, and achieve awesome throughput (almost identical to pure streaming). Problem: we're not even close to touching 100% of boot-only BBT, so (1) gang-loading takes more time than it should, (2) there's memory pressure and we hit the pagefile a lot. Bottom line: no gains. I ran my tests on a P5-100, 32MB, NT4, shitty SCSI disk, recent winword9 build. If you're interested, I have a bunch of charts and data on this (I spent a good portion of my life on this!). BTW, it would be nice if we went open-source within MS (or at least DAD and NT)-idea recommended by VinodV. It would have been much easier if I had NT sources at the time.
* Contiguous BBT sections (JonDe point 2) won't help much by themselves, but would work perfectly with my prefetcher. The problem is that reads from winword.exe are interrupted by reads from mso9.dll, system DLLs, reg accesses, etc. The bottom line is that avg I/O time during winword launch is almost = disk seek time! Unbelievable, but true.
* Putting up a mock-up of the app window (DarrylR's point) is a good idea, but usability tests have shown that users do not like this (or so they think; when focus group studies were conducted with the first Ford Taurus in the early '80s, users said they didn't like its new looks, but when Ford started selling them, people changed their minds; go figure). There were other proposals in Office, such as a rich splash screen, actually a welcome screen, with a progress bar and options: new file, new from template, open file (+ listbox), etc. Users rejected it too. You can check it out on http://office10/bin/tables.asp?docType=Prototype&sortBy=Date (check out my prototype while you're there--it's the 1st one on the list).
* Winword9 launch times on Hydra are amazing (but obvious): less than 2 s on wtsoff. Hydra offers tons of completely new

possibilities for mega-heavy optimizations.

Pls let me know what you think. thx

Arthur Zwiegincew
** Hardcore Computer Maniac **

-----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.

From: Steven Sinofsky
Sent: Thursday, January 21, 1999 11:49 PM
To: Jon DeVaan
Subject: RE: Applications boot time (1 s Word 1st-boot)

He works for jimw. qed.

Actually he's a new guy and he's incredibly hyper about stuff. I think he left me off of this one since after about 10 rounds over his radical idea for a new shell I urged him to focus on the here and now.

This is a strange idea.
-----Original Message-----
From: Jon DeVaan
Sent: Thursday, January 21, 1999 11:46 PM
To: Steven Sinofsky; Duane Campbell
Subject: FW: Applications boot time (1 s Word 1st-boot)

Who is this guy?

-----Original Message-----
From: Arthur Zwiegincew
Sent: Thursday, January 21, 1999 6:50 PM
To: Bill Gates; Jon DeVaan; Jim Allchin (Exchange)
Subject: RE: Applications boot time (1 s Word 1st-boot)

Re your wishes below: I'm from Office Perf and I've come up with an idea to radically speed up app launch and other scenarios. It's currently being patented (MS#116278.1). For it to succeed, we need NT's blessing (it requires some changes in VMM). I have solutions that would work with (1) Office/MS apps only, (2) any random app.

Bottom line:
Word9 launch on a clean system (i.e. registry in memory) with enough RAM (maybe 32MB) and a very slow disk (3MB/s), should take 0.32 s (streaming+decompression; see below) + CPU time (less than 0.5 s) + random I/O time + Explorer time ~= 1-1.5 s. Radical.

In short:
The idea is to instrument the page fault handler. Apps call MmBeginScenario(GUID, ...) and MmEndScenario(GUID) to tell VMM that they're executing common scenarios (optionally, in the future we could log all pf's and run idle pattern-matching to find scenarios w/o apps' involvement). VMM logs pf's as they occur, and then, at idle, all pages that were faulted are copied into a scenario file. The next time this scenario is run, VMM reads everything in one I/O and sets up PTEs/VADs for all pages in the scenario (we'll use NTFS defrag hooks to make sure this really is one disk I/O). When the pages are actually needed, they'll be soft-faulted into the app working set. This is the holy grail--we're reading exactly the set of EXE/DLL (incl. system DLLs) pages that we need. All remaining I/O is random stuff, like normal.dot, and the registry (which totally sucks, they need some big-time, major fixes to the registry; on a loaded system there are nearly as many registry I/Os as there are winword.exe I/Os!!! I have some ideas how to help Office re registry). There are a bunch of issues, but the ones I have thought about, I was able to resolve.

All other approaches I know of, either complement this (e.g. BBT), or are inferior (e.g. Tune-Up Wizard).

An early spec is here:
<< File: spec.doc >>
Please try to convince NT to spare some resources. If they can't, I'd be happy to write this stuff myself, but I would have to have DAD's blessing (like I said, I'm from Office).

Re stuff you mention below:
* Gang-loading from user mode (JonDe point 1) does work at disk-speed, even though we're issuing zillions of IRPs. As long as they come with a high-enough frequency, we can keep the disk spinning, and achieve awesome throughput (almost identical to pure streaming). Problem: we're not even close to touching 100% of boot-only BBT, so (1) gang-loading takes more time than it should, (2) there's memory pressure and we hit the pagefile a lot. Bottom line: no gains. I ran my tests on a P5-100, 32MB, NT4, shitty SCSI disk, recent winword9 build. If you're interested, I have a bunch of charts and data on this (I spent a good portion of my life on this!). BTW, it would be nice if we went open-source within MS (or at least DAD and NT)-idea recommended by VinodV. It would have been much easier if I had NT sources at the time.
* Contiguous BBT sections (JonDe point 2) won't help much by themselves, but would work perfectly with my prefetcher. The problem is that reads from winword.exe are interrupted by reads from mso9.dll, system DLLs, reg accesses, etc. The bottom

line is that avg I/O time during winword launch is almost = disk seek time! Unbelievable, but true.
* Putting up a mock-up of the app window (DarrylR's point) is a good idea, but usability tests have shown that users do not like this (or so they think; when focus group studies were conducted with the first Ford Taurus in the early '80s, users said they didn't like its new looks, but when Ford started selling them, people changed their minds; go figure). There were other proposals in Office, such as a rich splash screen, actually a welcome screen, with a progress bar and options: new file, new from template, open file (+ listbox), etc. Users rejected it too. You can check it out on <http://office10/bin/tables.asp?docType=Prototype&sortBy=Date> (check out my prototype while you're there--it's the 1st one on the list).
* Winword9 launch times on Hydra are amazing (but obvious): less than 2 s on wtsoff. Hydra offers tons of completely new possibilities for mega-heavy optimizations.

Pls let me know what you think. thx

Arthur Zwiegincew
** Hardcore Computer Maniac **

-----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.

From: Steven Sinofsky
Sent: Thursday, January 28, 1999 8:52 AM
To: Arthur Zwiegincew
Cc: Jim Walsh; Duane Campbell
Subject: RE: Applications boot time [time for Word is now down to 0.3 s]

I think it s great that you've got this idea. I think the next steps for us are to actually do it and see if it works the way we need it to work and the way you imagine in your mail. I think we've taken the mail thread as far as we can and we just need to be careful about setting folks' expectations.

Let's just get it done if we think it can be done!

-----Original Message-----
From: Arthur Zwiegincew
Sent: Thursday, January 28, 1999 2:25 AM
To: Bill Gates; Jim Walsh; Rick Rashid; Amitabh Srivastava
Cc: Eric Rudder; Jon DeVaan; Steven Sinofsky; Nathan Myhrvold; Mark Lucovsky
Subject: RE: Applications boot time [time for Word is now down to 0.3 s]

I don't mean to stir up this exchange any further, but I just wrote up a summary of what I have that you might find interesting. The upshot: the most state-of-the-art method I can imagine now, will decrease Word launch time to 0.3 s. I'm absolutely serious. It's in point 4.3 in the attached mail. It doesn't require any totally sweeping OS changes, and doesn't eat up memory. Pure gravy.

The document is long, but structured and easy-to-browse.

The most important stuff is in blue.

<< Message: The Quest for the Ultimate Prefetcher >>
Arthur Zwiegincew
** Hardcore Computer Maniac **
-----Original Message-----
From: Jim Walsh
Sent: Wednesday, January 27, 1999 10:49 AM
To: Rick Rashid; Bill Gates; Amitabh Srivastava; Arthur Zwiegincew
Cc: Eric Rudder; Jon DeVaan; Steven Sinofsky; Nathan Myhrvold; Mark Lucovsky
Subject: RE: Applications boot time (1 s Word 1st-boot)

This works great, obviously the tradeoff is giving up fixed memory pages which are fairly precious in most corporate machines (in which according to Intel it was only November '97 when 32Mb machines first started outselling 16Mb machines, with the installed base of course being even further behind).

A related idea we're currently prototyping as proof of concept is pre-loading pages with something like osa9.exe, but having it stay around as a thin process that tickles the pages on a regular basis (lest the OS discard them). It can be smarter - can only preload the pages from the BBT boot section(s), thus not taking up memory for pages not likely to be touched during boot. Again, you're taking a bit hit on other apps because of the memory you're using. Perhaps a feature that lets the user (or admin) explicitly choose this for particular apps. This idea can be further enhanced by, as you mention, compressing these pages, thus reducing the disk load time. It can also be smart about when to preload - the current prototype only does anything on 64Mb+ systems with 16Mb+ free. More research will let us adjust those numbers appropriately. This, or other preload techniques, are certainly worth investigating for high memory systems, or when users explicitly trade off memory for speed of particular apps (we're talking to an OEM about an 'Office computer' optimized for running Office, for example).

Another idea would be load and tickle, in-memory, the compressed pages, thus having compressed versions of only the pages you care about, and when the app launches a little bit of code uncompresses the pages in a manner exactly as if they'd all been paged in from disk. This takes less physical memory while 'waiting', but requires that the full amount be available at launch (or shortly thereafter), and with the decompression etc. actual memory demand at launch time would be somewhat higher.

We're currently breaking down all disk accesses during launch by component (including OS), and determining the theoretical gains for the different pieces, but for Winword it looks like winword.exe and mso9.dll comprise the vast majority of disk access during launch, so there's certainly promise with this or any other technique that reduces the disk access required to get code loaded off disk.

Jim

-----Original Message-----
From: Rick Rashid
Sent: Wednesday, January 27, 1999 9:48 AM
To: Bill Gates; Amitabh Srivastava; Arthur Zwiegincew
Cc: Eric Rudder; Jon DeVaan; Steven Sinofsky; Nathan Myhrvold; Mark Lucovsky; Jim Walsh
Subject: RE: Applications boot time (1 s Word 1st-boot)

On systems with enough RAM under Win98 you can allocate a ramdisk and put the core office files into it and get the same effect described below - nearly instant start times. I tried this on one of my machines with 128MB of RAM. I partitioned the ram into 64MB system and 64MB ramdisk. The ramdisk I used was a freeware DOS 6.x XMS tsr that I pulled off the Internet. I load the ramdisk with the relevant files during bootup in my autoexec.bat file from a temp directory and set my paths so that the ramdisk is the first one on the list. Works like a charm (although getting the right files together isn't trivial). Generally speaking, running Excel, Powerpoint and Word becomes a < 1sec task - more like what you get when you have the files "cached" from a previous run - largely independent of previous activity such as running Visual Studio builds.

Of course, you could optimize this quite a bit. You seem to need about 30MB for the "largest" of the exes and dlls (e.g. Winword.exe, excel.exe, mso97.dll). You could imagine a ramdisk that uses compression (we may already have one lying around but I couldn't find it in my 5 minute search of the Internet) and get a factor of two compression so you could probably get away with less than 15MB of actual RAM. A specialized "office" compressor could probably get a factor of 3. You could also "preload" the cache from a contiguous part of disk rather than getting it from normal DOS file copies.

I mentioned this to Jim Walsh. He said they did some experiments with ram caches a while back but he is going to check into it again. Of course, this isn't the most efficient way to use memory and an idea like this won't help a low memory system. The main thing that has changed is the increasing prevalence of large memory systems due to low memory prices.

Bob Fitzgerald did something along these lines for the NT team as an experiment to help them speed up boot. He built a RAM boot cache which preloads the physical disk blocks needed by NT during the idle times when NT is waiting for device probes to complete. Although they don't plan to ship it for schedule reasons, the NT team has used this code profitably to help find non-disk bottlenecks in their boot procedures.

-Rick

-----Original Message-----
From: Bill Gates
Sent: Sunday, January 24, 1999 5:48 PM
To: Amitabh Srivastava; Arthur Zwiegincew
Cc: Eric Rudder; Jon DeVaan; Steven Sinofsky; Nathan Myhrvold; Mark Lucovsky; Rick Rashid
Subject: FW: Applications boot time (1 s Word 1st-boot)

The value of applications being able to boot at this kind of speed is totally HUGE HUGE HUGE. This alone would get most users to upgrade. I want to totally understand ho we make this happen.

I do NOT understand why we would need any new hooks in NT to do log the file of pages we need. Vulcan should be able to gather all the information that Andrew thinks we need - the page fault list during the boot up time.

When you do an install of Word if you are not starved for disk space we would run a piece of boot that would set up the "boot file" and make sure to enable the code that uses the boot file.

There may need to be something in NT to have the boot file read and the memory map be set up the right way. Has anyone figure out what work would be required for this?
I think its worth doing so ASAP.

Am I missing something here where we really need to change NT more than just a special load?

Whenever I get mail like this I am reminded of how SUPRISED I am that no one ever suggests what we should do about the registry - where do we go to have something that is less of a problem in terms of management, deployment and speed. It blows the mind.

Getting app start up speed to be this fast would be super fantastic. Lets figure out how to make this reall

-----Original Message-----
From: Nathan Myhrvold
Sent: Sunday, January 24, 1999 4:18 PM

To: Bill Gates
Subject: FW: Applications boot time (1 s Word 1st-boot)

FYI - doing things like splitting DLLs then remerging another way, cloning small sections if need be is a great example of being able to manipulate a large source base more effectively with tools that with people.

Nathan

-----Original Message-----
From: Amitabh Srivastava
Sent: Sunday, January 24, 1999 9:28 AM
To: Nathan Myhrvold
Cc: Rick Rashid
Subject: RE: Applications boot time (1 s Word 1st-boot)

For Office, Arthur is right the behavior for Office is that we keep bouncing back and forth between ms09 and winword. There is little you can do. Our opinion is Office has a reverse situation from NT we need dll-splitting followed by dll-merging : MS09 is too big. We need to analyze MS09 and break it into few parts. The part which is specific for Word should be dll-merged with Word, the part specific to Excel should be dll-merged with Excel and so on. The part that is shared can be kept as a separate dll. It is also possible to clone critical pieces. Our hope is that with this approach we can stop the bouncing back and forth. We will do an analysis and determine what the structure should be.

We are trying to get a new version of BBT and Vulcan released to the company at the end of this month. NT has been able to optimize 50 more dlls over what they ever did before. Following this, We plan to study dll-merging and make if very effective. We'll be studying Office, SQL .. more closely -- our focus lately has been on NT. Regardless, all this will work with Arthur's scheme.

Amitabh

-----Original Message-----
From: Nathan Myhrvold
Sent: Friday, January 22, 1999 11:15 PM
To: Amitabh Srivastava
Cc: Rick Rashid
Subject: FW: Applications boot time (1 s Word 1st-boot)

-----Original Message-----
From: Bill Gates
Sent: Friday, January 22, 1999 1:39 PM
To: Eric Rudder; Nathan Myhrvold; Rick Rashid
Subject: FW: Applications boot time (1 s Word 1st-boot)

FYI....

-----Original Message-----
From: Arthur Zwiegincew
Sent: Thursday, January 21, 1999 6:50 PM
To: Bill Gates; Jon DeVaan; Jim Allchin (Exchange)
Subject: RE: Applications boot time (1 s Word 1st-boot)

Re your wishes below: I'm from Office Perf and I've come up with an idea to radically speed up app launch and other scenarios. It's currently being patented (MS#116278.1). For it to succeed, we need NT's blessing (it requires some changes in VMM). I have solutions that would work with (1) Office/MS apps only, (2) any random app.

Bottom line:
Word9 launch on a clean system (i.e. registry in memory) with enough RAM (maybe 32MB) and a very slow disk (3MB/s), should take 0.32 s (streaming+decompression; see below) + CPU time (less than 0.5 s) + random I/O time + Explorer time ~= 1-1.5 s. Radical.

In short:
The idea is to instrument the page fault handler. Apps call MmBeginScenario(GUID, ...) and MmEndScenario(GUID) to tell VMM that they're executing common scenarios (optionally, in the future we could log all pf's and run idle pattern-matching to find scenarios w/o apps' involvement). VMM logs pf's as they occur, and then, at idle, all pages that were faulted are copied into a

scenario file. The next time this scenario is run, VMM reads everything in one I/O and sets up PTEs/VADs for all pages in the scenario (we'll use NTFS defrag hooks to make sure this really is one disk I/O). When the pages are actually needed, they'll be soft-faulted into the app working set. This is the holy grail--we're reading exactly the set of EXE/DLL (incl. system DLLs) pages that we need. All remaining I/O is random stuff, like normal.dot, and the registry (which totally sucks, they need some big-time, major fixes to the registry; on a loaded system there are nearly as many registry I/Os as there are winword.exe I/Os!!! I have some ideas how to help Office re registry). There are a bunch of issues, but the ones I have thought about, I was able to resolve.

All other approaches I know of, either complement this (e.g. BBT), or are inferior (e.g. Tune-Up Wizard).

An early spec is here:
<< File: spec.doc >>
Please try to convince NT to spare some resources. If they can't, I'd be happy to write this stuff myself, but I would have to have DAD's blessing (like I said, I'm from Office).

Re stuff you mention below:
* Gang-loading from user mode (JonDe point 1) does work at disk-speed, even though we're issuing zillions of IRPs. As long as they come with a high-enough frequency, we can keep the disk spinning, and achieve awesome throughput (almost identical to pure streaming). Problem: we're not even close to touching 100% of boot-only BBT, so (1) gang-loading takes more time than it should, (2) there's memory pressure and we hit the pagefile a lot. Bottom line: no gains. I ran my tests on a P5-100, 32MB, NT4, shitty SCSI disk, recent winword9 build. If you're interested, I have a bunch of charts and data on this (I spent a good portion of my life on this!). BTW, it would be nice if we went open-source within MS (or at least DAD and NT)-idea recommended by VinodV. It would have been much easier if I had NT sources at the time.
* Contiguous BBT sections (JonDe point 2) won't help much by themselves, but would work perfectly with my prefetcher. The problem is that reads from winword.exe are interrupted by reads from mso9.dll, system DLLs, reg accesses, etc. The bottom line is that avg I/O time during winword launch is almost = disk seek time! Unbelievable, but true.
* Putting up a mock-up of the app window (DarrylR's point) is a good idea, but usability tests have shown that users do not like this (or so they think; when focus group studies were conducted with the first Ford Taurus in the early '80s, users said they didn't like its new looks, but when Ford started selling them, people changed their minds; go figure). There were other proposals in Office, such as a rich splash screen, actually a welcome screen, with a progress bar and options: new file, new from template, open file (+ listbox), etc. Users rejected it too. You can check it out on http://office10/bin/tables.asp?docType=Prototype&sortBy=Date (check out my prototype while you're there--it's the 1st one on the list).
* Winword9 launch times on Hydra are amazing (but obvious): less than 2 s on wtsoff. Hydra offers tons of completely new possibilities for mega-heavy optimizations.

Pls let me know what you think. thx

Arthur Zwiegincew
** Hardcore Computer Maniac **

-----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.