

Re: 32-bit programs on Windows x64

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2008-07/msg00321.html>

- *From:* "Peter Olcott" <NoSpam@xxxxxxxxxxxxxx>
 - *Date:* Sun, 6 Jul 2008 19:08:08 -0500
-

"Joseph M. Newcomer" <newcomer@xxxxxxxxxxxxxx> wrote in message news:tul274tvva6eful7vfvupff52lgig717qf@xxxxxxxxxxxx

See below...

On Sun, 6 Jul 2008 12:49:16 -0500, "Peter Olcott" <NoSpam@xxxxxxxxxxxxxx> wrote:

"Joseph M. Newcomer" <newcomer@xxxxxxxxxxxxxx> wrote in message news:2fu174pnif6e48hff9ss2p86ht06uug60p@xxxxxxxxxxxx

See below...

On Sun, 6 Jul 2008 09:50:32 -0500, "Peter Olcott" <NoSpam@xxxxxxxxxxxxxx> wrote:

"Joseph M. Newcomer" <newcomer@xxxxxxxxxxxxxx> wrote in message news:46ev645h0n75e8rgc04am76tbfe592c3@xxxxxxxxxxxx

See below...

On Sat, 5 Jul 2008 08:35:33 -0500, "Peter Olcott" <NoSpam@xxxxxxxxxxxxxx> wrote:

"Joseph M. Newcomer" <newcomer@xxxxxxxxxxxxxx> wrote in message news:f6nt64937d35g2tii20qh5sskmvqccp1pp@xxxxxxxxxxxx

Re: 32-bit programs on Windows x64

See
below...

On
Fri,
4
Jul
2008
11:36:34
-0500,
"Peter
Olcott"
<NoSpam@xxxxxxxxxxxxxx>
wrote:

"Joseph
M.
Newcomer"
<newcomer@xxxxxxxxxxxxxx>
wrote
in
message
<news:ntvq64djj32o6smjpgqj7soo397b09ilk@xxxxxxxx>

There
is
no
way
to
discover
this.
There
are
so
many
processes
running
that
if
you
really
need
4GB
of
physical
memory
available
to
hold

Re: 32-bit programs on Windows x64

your
entire
process,
you
will
need
at
least
an
8GB
machine,
AND
EVEN
THEN
THERE
IS
NO
GUARANTEE
THAT
PAGING
WILL
NOT
OCCUR.
It
is
essentially
not
under
your
control.
You
can
try
::VirtualLock,
but
it
doesn't
guarantee
that
you
can
lock
things
down
(and
you
need
special
admin
configuration
parameters

Re: 32-bit programs on Windows x64

to
allow
it
to
work
at
all,
and
you
have
to
set
your
working
set
quota).

Any
belief
that
Windows
can
sustain
real-time
behavior
is
based
on
various
delusional
systems.
It
is
not
real-time
and
any
illusion
that
it
is
real-time
is
just
that,
an
illusion.

A
swapping
operation

Re: 32-bit programs on Windows x64

takes
10–50ms
depending
on
a
variety
of
parameters.
But
it
is
impossible
to
process
4.2GB
of
data
in
1/10
of
a
second
using
ANY
algorithm,
even
if
there
is
nothing
going
on.
Therefore
the
only
thing
that
matters
is
the
actual
working
set.

My
system
requires
4
GB
to

Re: 32-bit programs on Windows x64

store
its
DFA
pixel
pattern
recognizer.
To
match
any
particular
pixel
pattern
only
requires
traversing
a
tiny
subset
of
this
4
GB
of
data.
The
4
GB
of
data
is
required
so
that
millions
of
different
pixel
patterns
can
be
concurrently
matched.

First,
note
that
you
are
never
going

Re: 32-bit programs on Windows x64

to
get
4GB,
and
in
fact
never
going
to
get
2GB,
on
a
standard
Win32
installation.
If
you
are
lucky,
you
might
get
1.5GB
on
a
good
day.
No
promises.

If
you
run
a
Win32
app
on
a
Win64
system,
you
will
not
get
4GB.
You
might
get,
under
ideal

Re: 32-bit programs on Windows x64

conditions,
about
3.5,
but
that
might
be
optimistic.
I
don't
know
how
the
basic
allocator
works
near
that
level
of
stress.

I think that
if enough
RAM is
available
you are
supposed
to
get the
whole
4.0GB.

YOU ARE STILL NOT
GETTING IT!!!! IF THERE
IS 512K ON
THE
MACHINE, YOU STILL
GET THE 4GB!!!
THE AMOUNT OF RAM
ON THE MACHINE IS
COMPLETELY,
TOTALLY
UNRELATED TO THE
AVAILABLE ADDRESS
SPACE!!!! HOW MANY
TIMES DO I HAVE TO
SAY THIS???

DO

Re: 32-bit programs on Windows x64

NOT CONSIDER
PHYSICAL RAM AS
IMPOSING ANY
LIMITATIONS ON
VIRTUAL ADDRESS
SPACE!!!

All
physical RAM does is give
you
more physical address
space. This can impact
performance,
but it has absolutely
NOTHING
to do with how much
address space is available to
you
in
a
process!!! Why do you
insist
on thinking about physical
memory as being an
operational
parameter in any of this?

What DOES matter here is
that you have to load, in
your
4GB of address space, all
the
32-bit DLLs that you are
using. These will be
scattered
around throughout your
address
space, cluttering it up,
fragmenting it, and
generally
interfering with your ability
to
get data space. Then there
has to be room for your
program code, constants,
and static
data, and that gets deducted.
The only difference
between
Win32 and Win64 is the
system
DLLS. In Win32, you also

Re: 32-bit programs on Windows x64

have KERNEL32.DLL,
USER32.DLL,
GDI32.DLL, NTDLL.DLL
and
ADVAPI32.DLL), these
interfaces are not present
when
you
are running 32-bit apps in
Win64
(I haven't verified the docs
are not lying about
this).
Surprise, surprise, your
executable and all these
DLLs TAKE UP
ADDRESS SPACE
which
makes it unavailable to you
for
data. So, on a good day, you
might be able to use
1.5GB
of the 2GB (Win32), 2.5GB
of the
3GB (server boot), or 3.5GB
of the 4GB (Win32 on
Win64).
If you're lucky. I may be
optimistic in these
estimates.
joe

3.5 GB may be enough, otherwise I can
always go with a
Win64
on Win64.

Concurrently
matched
sounds
suspicious.
How
many
threads
running

Re: 32-bit programs on Windows x64

on
how
many
processors
is
the
first
question
about
concurrency.
Every
thread
>
total
processors
slows
a
compute-bound
computation
down.

http://en.wikipedia.org/wiki/Deterministic_finite_state_machine

A DFA
inherently
does
concurrent
matching on
all of
its
patterns, as
opposed to
and contrast
with
making one
comparison
for each
pattern to be
matched.

No. A DFA/DFSM does
sequential matching of its
input
stream to the states; I saw
nothing in that article that
implies any concurrency.
If
you have several DFSMs,
and have

Re: 32-bit programs on Windows x64

created the union of them (a technique I learned over 40 years ago, and used to assign to beginning programming students), you first get a NDFSM, but you can always reduce an NDFSM to a DFSM, so you end up with a single DFSM that does sequential matching of input tokens to states.

When a DFA is used as a lexical analyzer, it does not need to compare its input string against every possible valid token string, one-at-a-time, the input string is simultaneously compared to all valid token strings in parallel. It does this with no actual comparisons at all, its all a matter of indexes into the state transition matrix.

Because of this the extra time required when additional strings are added to the set of possible valid token strings is zero. It takes the same amount of time (using a DFA) to compare one string against one string, as it does to compare one string against 10,000 strings, because the strings are compared in parallel rather than sequentially.

Re: 32-bit programs on Windows x64

It compares the string, sequentially, one token at a time, to the input string. The fact that the implementation is a state transition table does not make it concurrent.

Note that state transition tables are usually sparse matrices, and as such, are subject to some really interesting optimizations, such as "comb vector" layout, which substantially reduces the space required while maintaining the constant-time lookup.

joe

There are two different levels of abstraction here, physical implementation, and functional end-result. Even though the process is physically implemented as a sequential set of steps, the functional end-result of this sequential set of steps is that one string is compared against many strings at-the-same-time, simultaneously, concurrently, and in parallel.

I find the use of "parallel" in single-threaded code to be a very suspicious use of the word. If there is no concurrency, there is no parallelism.

This is proven by the fact that adding strings to be compared against does not add any time at all to the matching process.

No, that's just an FSM. No parallelism is involved

In other words you are right at one level of abstraction, and I am right at another different level of abstraction.

I don't see any concurrency at all. It's just an FSM.

joe

You are still stuck in a single level of abstraction.

So
it
then
boils
down
to
how
many
patterns/sec
you
can
match.
Note
that
the
L2
cache
is
probably
going
to
cost
you
an
order
of
magnitude
performance
already.
joe

Why should
it cost that
much? That
seems
unreasonably
high.
800 mhz

Re: 32-bit programs on Windows x64

RAM
should (at
least in
theory) be
no more
than
300%
slower than
a 2400 mhz
processor,
and the
purpose of
cache
is to reduce
this penalty,
not increase
it.

It may seem high, but in fact
if you end up thrashing
the
L2 cache, you will lose
10x-20x
performance. This has been
documented in a variety of
experiments over the years.
An L2
cache miss will cost you
10x-20x over an L2 cache
hit.
The lower bound is if all the
data
in the cache is unmodified;
the higher one is if there
is
a dirty cache line and it has
to
be written back first.
joe

<http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO>

Re: 32-bit programs on Windows x64

Perhaps
you
really
have
an
infeasible
project.
joe

On
Thu,
3
Jul
2008
14:03:01
-0700
(PDT),
PeteOlcott
<PeteOlcott@xxxxxxxx>
wrote:

What
part
of
"virtual
memory
is
not
physical
memory"
are
you
missing.
I've
said
this
eight
or
ten
or
a
dozen
times,
VIRTUAL
MEMORY
IS
NOT
PHYSICAL

MEMORY.

Not
only
does
my
process
require
4.0
GB
of
RAM,
it
is
also
time
critical.
If
there
is
any
swapping
to
and
from
VM,
then
the
current
project
becomes
infeasible.
I
must
know
how
much
actual
RAM
is
available
to
my
process,
even
though
this
actual
RAM
will

Re: 32-bit programs on Windows x64

be
mapped
to
VM.
If
three
is
any
swapping
to
and
from
disk,
the
current
project
becomes
infeasible,
my
maximum
response
time
is
1/10
second.

Joseph
M.
Newcomer
[MVP]
email:
newcomer@xxxxxxxxxxxxx
Web:
<http://www.flounder.com>
MVP
Tips:
http://www.flounder.com/mvp_tips.htm

Joseph
M.
Newcomer
[MVP]
email:
newcomer@xxxxxxxxxxxxx
Web:
<http://www.flounder.com>
MVP
Tips:
http://www.flounder.com/mvp_tips.htm

Re: 32-bit programs on Windows x64

Joseph M. Newcomer
[MVP]
email:
newcomer@xxxxxxxxxxxxx
Web:
<http://www.flounder.com>
MVP Tips:
http://www.flounder.com/mvp_tips.htm

Joseph M. Newcomer [MVP]
email: newcomer@xxxxxxxxxxxxx
Web: <http://www.flounder.com>
MVP Tips: http://www.flounder.com/mvp_tips.htm

Joseph M. Newcomer [MVP]
email: newcomer@xxxxxxxxxxxxx
Web: <http://www.flounder.com>
MVP Tips: http://www.flounder.com/mvp_tips.htm