Project **EaRing**: A Meta-Literal Machine

Zed A. Shaw

zedshaw@zedshaw.com

June 9, 2008

A Meta-Literal Machine

- **Everyone gets to announce a VM these days**

A Meta-Literal Machine

- Everyone gets to announce a VM these days
- **Their VMs are slow, because they're virtual**

A Meta-Literal Machine

- Everyone gets to announce a VM these days
- Their VMs are slow, because they're virtual
- **EaRing is literal, so it is as fast as your CPU**

A Meta-Literal Machine

- Everyone gets to announce a VM these days
- Their VMs are slow, because they're virtual
- EaRing is literal, so it is as fast as your CPU
- **EaRing sounds like Erlang so it's fast**

A Meta-Literal Machine

- Everyone gets to announce a VM these days
- Their VMs are slow, because they're virtual
- EaRing is literal, so it is as fast as your CPU
- EaRing sounds like Erlang so it's fast
- **Just like anything with memcache is fast**

A Meta-Literal Machine

- Everyone gets to announce a VM these days
- Their VMs are slow, because they're virtual
- EaRing is literal, so it is as fast as your CPU
- EaRing sounds like Erlang so it's fast
- Just like anything with memcache is fast
- **Or, anything with "cloud" computing**

A Meta-Literal Machine

- Everyone gets to announce a VM these days
- Their VMs are slow, because they're virtual
- EaRing is literal, so it is as fast as your CPU
- EaRing sounds like Erlang so it's fast
- Just like anything with memcache is fast
- Or, anything with "cloud" computing
- **So EaRing will be fast, fast, fast!**

What Is A Literal Machine?

- **A VM is a big while-loop**

What Is A Literal Machine?

- A VM is a big while-loop
- **Reading bytes**

What Is A Literal Machine?

- A VM is a big while-loop
- Reading bytes
- **And then running another language**

What Is A Literal Machine?

- A VM is a big while-loop
- Reading bytes
- And then running another language
- **Like C, or Ruby, or C-Ruby, or Scheme**

What Is A Literal Machine?

- A VM is a big while-loop
- Reading bytes
- And then running another language
- Like C, or Ruby, or C-Ruby, or Scheme
- **Or, Assembler**

A Literal Machine...

- **Makes the real byte codes**

A Literal Machine...

- Makes the real byte codes
- **You might call it an "assembler"**

A Literal Machine...

- Makes the real byte codes
- You might call it an "assembler"
- **They are pretty handy**

A Literal Machine...

- Makes the real byte codes
- You might call it an "assembler"
- They are pretty handy
- **For, you know, making fast code**

EaRing Is Meta

- **Meta is awesome**

EaRing Is Meta

- Meta is awesome
- **It lets people do amazing things**

EaRing Is Meta

- Meta is awesome
- It lets people do amazing things
- **Where would Rails 50k lines of code be...**

EaRing Is Meta

- Meta is awesome
- It lets people do amazing things
- Where would Rails 50k lines of code be...
- **Without Meta Meta Meta!**

EaRing Is Meta

- Meta is awesome
- It lets people do amazing things
- Where would Rails 50k lines of code be...
- Without Meta Meta Meta!
- **(Probably 20k still)**

EaRing Took Me Two Weeks!

- I am a god

EaRing Took Me Two Weeks!

- I am a god
- **Just like Avi and Gemstone**

EaRing Took Me Two Weeks!

- I am a god
- Just like Avi and Gemstone
- **I totally didn't cheat either**

EaRing Took Me Two Weeks!

- I am a god
- Just like Avi and Gemstone
- I totally didn't cheat either
- **Just like Avi and Gemstone**

EaRing Is For You

- **With EaRing, Even you can announce a new Ruby VM in a few short weeks.**

EaRing Is For You

- With EaRing, Even **you** can announce a new Ruby VM in a few short weeks.
- **Just take what I show you here today,**

EaRing Is For You

- With EaRing, Even **you** can announce a new Ruby VM in a few short weeks.
- Just take what I show you here today,
- **add any Ruby parser you can find,**

EaRing Is For You

- With EaRing, Even **you** can announce a new Ruby VM in a few short weeks.
- Just take what I show you here today,
- add any Ruby parser you can find,
- **and PRESTO, you are a player baby.**

EaRing

- **The Perfect Accessory To Every Ruby On Rails Company(tm)**

What Is EaRing Really?

- **Simple**

What Is EaRing Really?

- Simple
- **A parser for a simple assembly language**

What Is EaRing Really?

- Simple
- A parser for a simple assembly language
- **That generates machine code dynamically**

What Is EaRing Really?

- Simple
- A parser for a simple assembly language
- That generates machine code dynamically
- **Giving you dynamic access to functions**

What Is EaRing Really?

- Simple
- A parser for a simple assembly language
- That generates machine code dynamically
- Giving you dynamic access to functions
- **Doing this on multiple CPUs (x86-32, x86-64, PPC, SPARC)**

What Is EaRing Really?

- Simple
- A parser for a simple assembly language
- That generates machine code dynamically
- Giving you dynamic access to functions
- Doing this on multiple CPUs (x86-32, x86-64, PPC, SPARC)
- **From the same source**

Fibonacci Demo

- **The Best Benchmark Ever**

Fibonacci Demo

- The Best Benchmark Ever
- **There's No Way To Make It Faster**

Fibonacci Demo

- The Best Benchmark Ever
- There's No Way To Make It Faster
- **Every new VM announce uses it**

Fibonacci Demo

- The Best Benchmark Ever
- There's No Way To Make It Faster
- Every new VM announce uses it
- **Mine will blow the doors off**

Basic Iterative Fib In EaRing

- **This is so fast!**

Basic Iterative Fib In EaRing

- This is so fast!
- **Only 1 slide of code!**

```
function fibit(in : ulong) : ulong
    getarg.ul(R2, in)
    movi.ul(R1, 1)
    blti.ul(exit:, R2, 2)
    subi.ul(R2, R2, 1)
    movi.ul(R0, 1)
loop:
    subi.ul(R2, R2, 1)
    addr.ul(V0, R0, R1)
    movr.ul(R0, R1)
    addi.ul(R1, V0, 1)
    bnei.ul(loop:, R2, 0)
exit:
    movr.ul(RET, R1)
    ret
end
```

The Main Method

- **Notice the import. That's dynamic!**

```
%import("includes/fibit.asm", "fibit")

function main() : ulong
    prepare(1)
    movi.ul(R0, 93)

    pusharg.ul(R0)
    finish(fibit.fibit)
    retval.ul(R0)

    movr.ul(RET, R0)
    ret
end
```

Timing To Parse And Run?

- **Actually too fast to measure.**

Timing To Parse And Run?

- Actually too fast to measure.
- **About 0.005 seconds on most machines.**

```
$ time ./earing includes/fibit_main.asm main
EaRing.  Copyright 2008 Zed A. Shaw.
Done compiling includes/fibit_main.asm.  Enter ? to get t
main >>  2587060292317343101

real        0m0.005s
user        0m0.000s
sys         0m0.003s
```

Dynamic REPL

- **Yes, EaRing has a REPL**

Dynamic REPL

- Yes, EaRing has a REPL
- **It makes learning ASM pretty fun**

```
EaRing.  Copyright 2008 Zed A. Shaw.
Done compiling simpler.asm.  Enter ? to get the function
>>> main
>>  -0.839072
>>> ?
data_sharing_test: 57 bytes of code, 0 params defined
fibit_huge_loop: 68 bytes of code, 0 params defined
fibit: 77 bytes of code, 1 params defined
incrementer: 32 bytes of code, 0 params defined
main: 163 bytes of code, 0 params defined
make_test1_test2: 56 bytes of code, 0 params defined
nfibs: 117 bytes of code, 1 params defined
print_fibit10: 52 bytes of code, 0 params defined
print_nfibs10: 52 bytes of code, 0 params defined
print_reflect: 52 bytes of code, 0 params defined
puts_sample: 98 bytes of code, 0 params defined
ram_test: 83 bytes of code, 0 params defined
```

```
reflect: 18 bytes of code, 1 params defined
test2: 21 bytes of code, 0 params defined
test1: 29 bytes of code, 0 params defined
>>> test1
>>  100
>>> test2
>>  100
>>>
```

How Fast Is It?!

- **Let's do 5 million fib 93 calls!**

```
%import("includes/fibit.asm", "fibit")

function main() : ulong
    movi.ui(V1, 5000000)

again:
    prepare(1)
    movi.ul(R0, 93)
    pusharg.ul(R0)
    finish(fibit.fibit)
    retval.ul(R0)

    # decrement and repeat
    subi.ui(V1, V1, 1)
    bgti.ui(again:, V1, 0)

    # grab the last value as a check
```

```
    movr.ul(RET, R0)
    ret
end
```

Timing To Parse And Run?

- **Remember, this is 5.000.000 function calls**

Timing To Parse And Run?

- Remember, this is 5.000.000 function calls
- **To a Fibonacci sequence up to 93**

Timing To Parse And Run?

- Remember, this is 5.000.000 function calls
- To a Fibonacci sequence up to 93
- **About 1.8 seconds on my laptop**

```
$ time ./earing includes/fibit_huge.asm main
EaRing.  Copyright 2008 Zed A. Shaw.
Done compiling includes/fibit_huge.asm.  Enter ? to get t
main >>  2587060292317343101

real        0m1.755s
user        0m1.747s
sys         0m0.003s
```

More Features!

- **Fully dynamic functions.**

More Features!

- Fully dynamic functions.
- **Thread safe (hopefully) and library capable.**

More Features!

- Fully dynamic functions.
- Thread safe (hopefully) and library capable.
- **Can even access .so libraries!**

```
%library("/lib/libc.so.6", "libc")

function puts_sample() : int
    MY_NAME = "Zed A. Shaw"

    movi.p(R0, MY_NAME)
    prepare(1)
    pusharg.p(R0)
    finish(libc.puts)

    ret
end
```

The Results Of Puts

- **Yep, does what you think.**

```
EaRing.  Copyright 2008 Zed A. Shaw.
Done compiling includes/puts_sample.asm.  Enter ? to get
puts_sample Zed A. Shaw
>>  12
```

More Features!

- **You want more?!**

More Features!

- You want more?!
- **Dynamic constants!**

More Features!

- You want more?!
- Dynamic constants!
- **Reference Constants in other functions!**

```
%library("/lib/libc.so.6", "libc")

SOME_STUFF = "Hello!"

function i_have_constants() : void
    A_NAME = "Hi There"
    ret
end

function puts_other() : int
    movi.p(R0, self.i_have_constants.A_NAME)
    prepare(1)
    pusharg.p(R0)
    finish(libc.puts)

    movi.p(R0, SOME_STUFF)
    prepare(1)
```

```
    pusharg.p(R0)
    finish(libc.puts)

    ret
end
```

MORE FEATURES?!

- **Functions Are Also Dynamic Constants!**

MORE FEATURES?!

- Functions Are Also Dynamic Constants!
- **ONE FUNCTION CAN ALTER ANOTHER!**

```
function test1() : int
    movi.i(R1, 10000)
    movi.i(R0, 20)

    mulr.i(RET, R1, R0)
    ret
end

function test2(): int
    movi.i(RET, 100)
    ret
end

function make_test1_test2() : int
    movi.ui(R1, 0)

next:
```

```
    ldxi.c(R0, R1, self.test2)
    stxi.c(self.test1, R1, R0)
    addi.ui(R1, R1, 1)
    blti.ui(next:, R1, 21)

    calli(self.test1)

    ret
end
```

Oh That's Evil

- **But does it work?**

Oh That's Evil

- But does it work?
- **Of course!**

```
EaRing.  Copyright 2008 Zed A. Shaw.
Done compiling includes/func_swap.asm.  Enter ? to get th
>>> ?
make_test1_test2: 56 bytes of code, 0 params defined
test2: 21 bytes of code, 0 params defined
test1: 29 bytes of code, 0 params defined
>>> test1
>>  200000
>>> test2
>>  100
>>> make_test1_test2
>>  100
>>>
```

What About Garbage Collection?

- **Collection? Garbage?**

What About Garbage Collection?

- Collection? Garbage?
- **This is assembler baby.**

What About Garbage Collection?

- Collection? Garbage?
- This is assembler baby.
- **You do your own GC.**

```
%library("/lib/libc.so.6", "libc")

function ram_test() : void
    movi.ui(R0, 1024)
    prepare(1)
    pusharg.ui(R0)
    finish(libc.malloc)
    movr.ui(R0, RET)

    prepare(1)
    pusharg.ui(R0)
    finish(libc.free)

    ret
end
```

Any Dynamic Library Dynamically

- **You want GC? You got GC.**

Any Dynamic Library Dynamically

- You want GC? You got GC.
- **You want threads? You got threads.**

Any Dynamic Library Dynamically

- You want GC? You got GC.
- You want threads? You got threads.
- **You want printf?**

Any Dynamic Library Dynamically

- You want GC? You got GC.
- You want threads? You got threads.
- You want printf?
- **Sorry, uh, printf is weird.**

MORE FEATURES?!

- **Actual Real Very Nice Error Messages**

MORE FEATURES?!

- Actual Real Very Nice Error Messages
- **With, Line numbers even!**

MORE FEATURES?!

- Actual Real Very Nice Error Messages
- With, Line numbers even!
- **Already Better Than MRI**

```
function badoptype : v
    movi.blah(R0,R1)
end

function badfunctype : not
    movi.ui(R0,1)
end

function unterminated : v

function recovers : v
end

function badtokens : i
    movi.ui($$,$$)
end
```

Error Reporting Is Sweet

- **When I run that, it will make you cry.**

```
error.asm:1: error at COLON token in
[ FUNCTION IDENT | COLON ] unexpected ':'
error.asm:1: error at TYPE token in
[ | TYPE ] unexpected 'v'
error.asm:2: error at OP token in
[ statements | OP ] unexpected 'movi'
error.asm:5: error at COLON token in
[ statements FUNCTION IDENT | COLON ] unexpected ':'
error.asm:9: error at COLON token in
[ statements FUNCTION IDENT | COLON ] unexpected ':'
error.asm:12: error at COLON token in
[ statements FUNCTION IDENT | COLON ] unexpected ':'
error.asm:16: error at COLON token in
[ statements FUNCTION IDENT | COLON ] unexpected ':'
error.asm:17: invalid character '$'
```

It Wasn't That Hard

- **When I say, "two weeks", I'm not bragging.**

It Wasn't That Hard

- When I say, "two weeks", I'm not bragging.
- **I'm saying you could do it too.**

It Wasn't That Hard

- When I say, "two weeks", I'm not bragging.
- I'm saying you could do it too.
- **If you take the code I wrote (not generated)**

It Wasn't That Hard

- When I say, "two weeks", I'm not bragging.
- I'm saying you could do it too.
- If you take the code I wrote (not generated)
- **2001 lines of text.**

```
329 src/data.c
164 src/directives.c
111 src/earing.c
 36 src/error.c
216 src/module.c
 79 src/naming.c
184 src/ops.c
 14 src/repl.c
 75 src/util.c

166 src/data.h
 10 src/directives.h
  9 src/error.h
 20 src/grammar.h
 10 src/lexer.h
 20 src/naming.h
145 src/ops.h
```

```
      9 src/repl.h
     14 src/tokenize.h
      9 src/util.h

    150 src/grammar.y
    231 src/lexer.rl
Total: 2001
```

Now Leverage Existing Libraries

- **sglib for data structures I needed.**

Now Leverage Existing Libraries

- sglib for data structures I needed.
- **GNU lightning for generating the machine code.**

Now Leverage Existing Libraries

- sglib for data structures I needed.
- GNU lightning for generating the machine code.
- **Lemon Parser for making the parser. Here's a sample.**

```
module ::= statements.
parameters ::= LPAREN args RPAREN.
    parameters ::=.
args ::= args COMMA expr.
    args ::= expr.
    args ::=.
operation ::= OP DOT TYPE parameters.
    operation ::= OP parameters.
expr ::= HEX.
    expr ::= FLOAT.  expr ::= INT.
    expr ::= STR.  expr ::= CHR.
    expr ::= REG.  expr ::= LABEL.
function ::= FUNCTION function_decl block END.
function_decl ::= IDENT LPAREN function_params
                  RPAREN COLON TYPE.
```

Ragel For Lexer

- **Ragel is an awesome tool making this pretty easy.**

Ragel For Lexer

- Ragel is an awesome tool making this pretty easy.
- **Does a lot more work than most lexers.**

Then A Bit Of Magic

- **Ternary Search Trees**

Then A Bit Of Magic

- Ternary Search Trees
- **My Favorite Data Structure**

Then A Bit Of Magic

- Ternary Search Trees
- My Favorite Data Structure
- **Nearly impossible to explain**

Then A Bit Of Magic

- Ternary Search Trees
- My Favorite Data Structure
- Nearly impossible to explain
- **Fast as hell for looking up strings**

```
void *Naming_search(tnode *root, const char *s, int len)
{
    tnode *p = root;
    int i = 0;

    while(i < len && p) {
        if (s[i] < p->splitchar) {
            p = p->low;
        } else if (s[i] == p->splitchar) {
            i++;
            if(i < len) p = p->equal;
        } else {
            p = p->high;
        }
    }

    if(p) {
```

```
        return p->value;
    } else {
        return NULL;
    }
}

tnode *Naming_insert(tnode *p, const char *s, int len, vo
{
    if (p == NULL) {
        p = (tnode *) calloc(1,sizeof(tnode));
        p->splitchar = *s;
    }

    if (*s < p->splitchar) {
        p->low = Naming_insert(p->low, s, len, value);
    } else if (*s == p->splitchar) {
        if (len > 1) {
```

```
            // not done yet, keep going but one less
            p->equal = Naming_insert(p->equal, s+1, len -
        } else {
            p->value = value;
        }
    } else {
        p->high = Naming_insert(p->high, s, len, value);
    }

    return p;
}

void Naming_traverse(tnode *p, Naming_traverse_cb cb, voi
{
    if (!p) return;
    if(p->low) Naming_traverse(p->low, cb, data);
```

```
    if (p->equal) {
        Naming_traverse(p->equal, cb, data);
    }

    if(p->high) Naming_traverse(p->high, cb, data);

    if(p->value) cb(p->value, data);
}
```

Naming And TSTs

- **Trading space for speed**

Naming And TSTs

- Trading space for speed
- **Don't scale up to giant strings**

Naming And TSTs

- Trading space for speed
- Don't scale up to giant strings
- **But super fast for small strings**

Naming And TSTs

- Trading space for speed
- Don't scale up to giant strings
- But super fast for small strings
- **Especially language identifiers**

Naming And TSTs

- Trading space for speed
- Don't scale up to giant strings
- But super fast for small strings
- Especially language identifiers
- **Works like a Hashmap with extra features**

TST Extra Features

- **Can do a partial match (like begins with, ends with)**

TST Extra Features

- Can do a partial match (like begins with, ends with)
- **Can do most Regex over it**

TST Extra Features

- Can do a partial match (like begins with, ends with)
- Can do most Regex over it
- **Faster to not find a member**

TST Extra Features

- Can do a partial match (like begins with, ends with)
- Can do most Regex over it
- Faster to **not** find a member
- **Can do probable matches.**

TST Extra Features

- Can do a partial match (like begins with, ends with)
- Can do most Regex over it
- Faster to **not** find a member
- Can do probable matches.
- **Can't find 'idint' but there is 'ident'.**

TST Extra Features

- Can do a partial match (like begins with, ends with)
- Can do most Regex over it
- Faster to **not** find a member
- Can do probable matches.
- Can't find 'idint' but there is 'ident'.
- **All great things when writing a parser**

Now, Put It All Together

- **An assembler,**

Now, Put It All Together

- An assembler,
- **That parses a real-ish language,**

Now, Put It All Together

- An assembler,
- That parses a real-ish language,
- **With error messages and fast parsing,**

Now, Put It All Together

- An assembler,
- That parses a real-ish language,
- With error messages and fast parsing,
- **Which can generate actual machine code,**

Now, Put It All Together

- An assembler,
- That parses a real-ish language,
- With error messages and fast parsing,
- Which can generate actual machine code,
- **With introspection, dynamic libraries, imports,**

Now, Put It All Together

- An assembler,
- That parses a real-ish language,
- With error messages and fast parsing,
- Which can generate actual machine code,
- With introspection, dynamic libraries, imports,
- **Yet you can modify and manipulate the results,**

Now, Put It All Together

- An assembler,
- That parses a real-ish language,
- With error messages and fast parsing,
- Which can generate actual machine code,
- With introspection, dynamic libraries, imports,
- Yet you can modify and manipulate the results,
- **And, on multiple CPUs from one source.**

100 Days For MagLev?!

- **Two Weeks! I win!**

100 Days For MagLev?!

- Two Weeks! I win!
- **Yeah yeah, they got an Object Database.**

100 Days For MagLev?!

- Two Weeks! I win!
- Yeah yeah, they got an Object Database.
- **Those have been real successful.**

Why You Should Buy EaRing

- **Buy?! What The FUCK I DEMAND OPEN SOURCE.**

Fine, It's Open Source Then

- **Sales Job Done!**

Disclaimer

- **Zed A. Shaw and the EaRing project are not responsible for any limitations, effects, or implications of using said software not limited to but including hair loss, nuclear reactor failures, performance limitations, difficult debugging, infection of the right eye, deterioration of small intestines, or sudden rapid blindness.**

The Bonus To You

- **EaRing will be the perfect accessory to any Ruby company.**

The Bonus To You

- EaRing will be the perfect accessory to any Ruby company.
- **To be top in the Ruby world, you need a VM.**

The Bonus To You

- EaRing will be the perfect accessory to any Ruby company.
- To be top in the Ruby world, you need a VM.
- **Without one, you're just a "Rails Shop". That's lame.**

The Bonus To You

- EaRing will be the perfect accessory to any Ruby company.
- To be top in the Ruby world, you need a VM.
- Without one, you're just a "Rails Shop". That's lame.
- **With EaRing, you have 50% of what you need to make a Ruby VM good enough to present.**

Imagine The Press Releases!

- **The Blogosphere will be blowing you harder than a 2 dollar whore with a bag of cash and no teeth.**

Imagine The Press Releases!

- The Blogosphere will be blowing you harder than a 2 dollar whore with a bag of cash and no teeth.
- **Imagine, all those Ruby programmers who blog so much, all the time, rather than, you know, writing good code.**

Imagine The Press Releases!

- The Blogosphere will be blowing you harder than a 2 dollar whore with a bag of cash and no teeth.
- Imagine, all those Ruby programmers who blog so much, all the time, rather than, you know, writing good code.
- **How influential will their pretty 5 paragraph posts about your VM be to the world?**

Imagine The Press Releases!

- The Blogosphere will be blowing you harder than a 2 dollar whore with a bag of cash and no teeth.
- Imagine, all those Ruby programmers who blog so much, all the time, rather than, you know, writing good code.
- How influential will their pretty 5 paragraph posts about **your** VM be to the world?
- **They're Ruby blogs! Your grandma will know! Senators will know! The WORLD will know!**

Resurrect Failing Platforms

- **Got a VM that only 10 companies pay for like Gemstone?**

Resurrect Failing Platforms

- Got a VM that only 10 companies pay for like Gemstone?
- **Got a platform that is dying faster than Darfur like Sun?**

Resurrect Failing Platforms

- Got a VM that only 10 companies pay for like Gemstone?
- Got a platform that is dying faster than Darfur like Sun?
- **Got a reputation for raping children like Microsoft?**

Resurrect Failing Platforms

- Got a VM that only 10 companies pay for like Gemstone?
- Got a platform that is dying faster than Darfur like Sun?
- Got a reputation for raping children like Microsoft?
- **EaRing will let you resurrect your failing platform, to the applause of millions!**

Resurrect Failing Platforms

- Got a VM that only 10 companies pay for like Gemstone?
- Got a platform that is dying faster than Darfur like Sun?
- Got a reputation for raping children like Microsoft?
- EaRing will let you resurrect your failing platform, to the applause of millions!
- **No, billions!**

Be A Good Guy

- **You're helping open source!**

Be A Good Guy

- You're helping open source!
- **How could that possibly go wrong?**

Be A Good Guy

- You're helping open source!
- How could that possibly go wrong?
- **Sun and Microsoft have great histories with open source.**

Be A Good Guy

- You're helping open source!
- How could that possibly go wrong?
- Sun and Microsoft have **great** histories with open source.
- **Open source projects like NetBSD, JetSpeed, JBoss, and Apache have never been taken over and exploited by big companies.**

Be A Good Guy

- You're helping open source!
- How could that possibly go wrong?
- Sun and Microsoft have **great** histories with open source.
- Open source projects like NetBSD, JetSpeed, JBoss, and Apache have **never** been taken over and exploited by big companies.
- **Nothing will ever go wrong with this setup.**

Be A Good Guy

- You're helping open source!
- How could that possibly go wrong?
- Sun and Microsoft have **great** histories with open source.
- Open source projects like NetBSD, JetSpeed, JBoss, and Apache have **never** been taken over and exploited by big companies.
- Nothing will ever go wrong with this setup.
- **If you help with EaRing, even you could be one of the great benefactors of the open source world.**

Be A Good Guy

- You're helping open source!
- How could that possibly go wrong?
- Sun and Microsoft have **great** histories with open source.
- Open source projects like NetBSD, JetSpeed, JBoss, and Apache have **never** been taken over and exploited by big companies.
- Nothing will ever go wrong with this setup.
- If you help with EaRing, even you could be one of the great benefactors of the open source world.
- **I promise that EaRing will always be free and open.**

Be A Good Guy

- You're helping open source!
- How could that possibly go wrong?
- Sun and Microsoft have **great** histories with open source.
- Open source projects like NetBSD, JetSpeed, JBoss, and Apache have **never** been taken over and exploited by big companies.
- Nothing will ever go wrong with this setup.
- If you help with EaRing, even you could be one of the great benefactors of the open source world.
- I promise that EaRing will always be free and open.
- **That's been so successful for me in the past.**

Amdahl's Law

- 1/((1-f)+(f/p))

Amdahl's Law

- 1/((1-f)+(f/p))
- **Anyone know what this is?**

Amdahl's Law

- 1/((1-f)+(f/p))
- Anyone know what this is?
- **It's mathematics for, "You're Fucked".**

Amdahl's Law

- 1/((1-f)+(f/p))
- Anyone know what this is?
- It's mathematics for, "You're Fucked".
- **f == Fraction of a program you can parallelize.**

Amdahl's Law

- 1/((1-f)+(f/p))
- Anyone know what this is?
- It's mathematics for, "You're Fucked".
- f == Fraction of a program you can parallelize.
- **p == Number of CPUs you can run it on in parallel.**

Amdahl's Law And Programmers

- **Imagine that programmers aren't very good.**

Amdahl's Law And Programmers

- Imagine that programmers aren't very good.
- **I know, it's a stretch.**

Amdahl's Law And Programmers

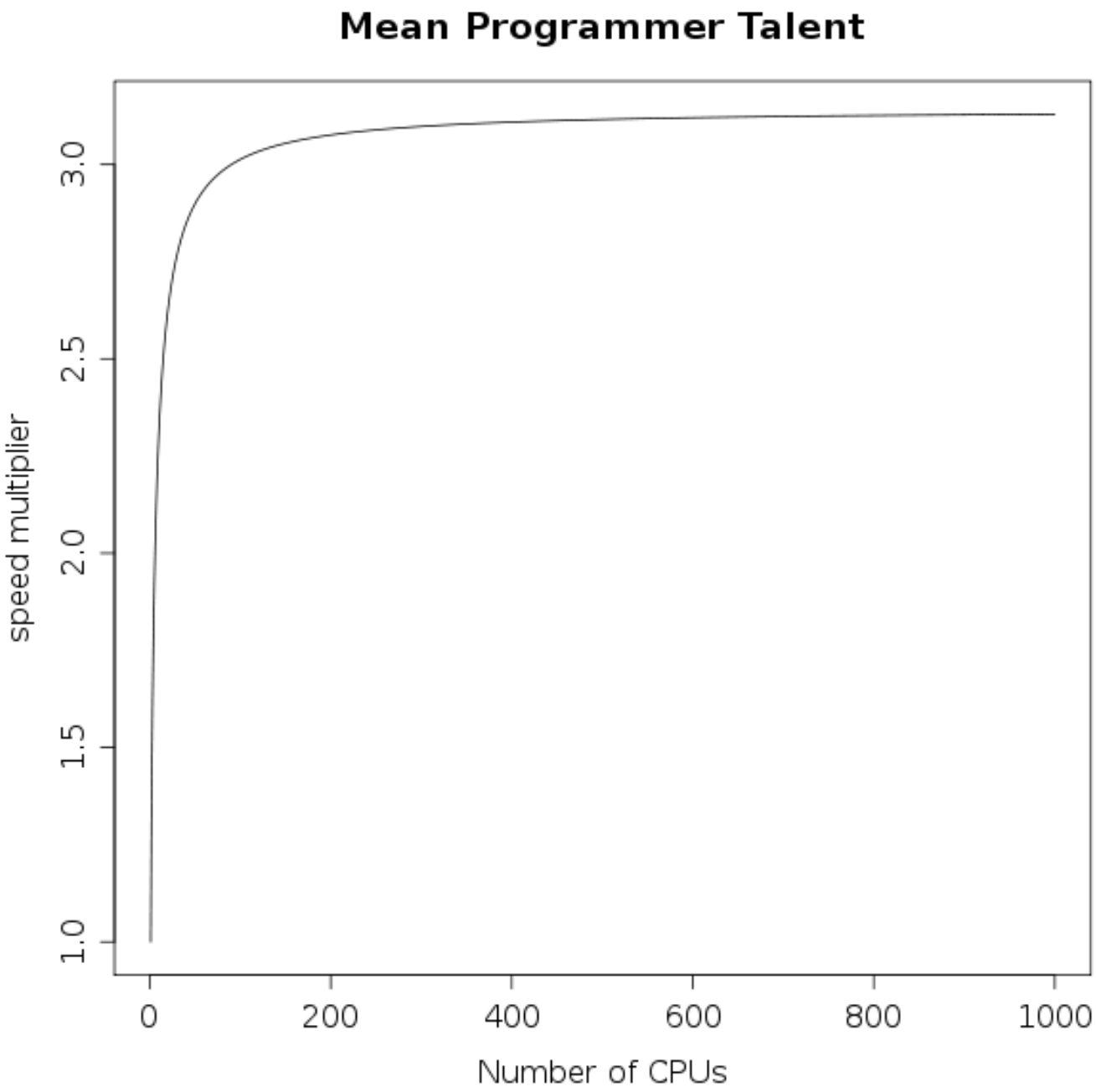
- Imagine that programmers aren't very good.
- I know, it's a stretch.
- **Now, imagine that there's a range of values for f depending on how good they can parallelize.**

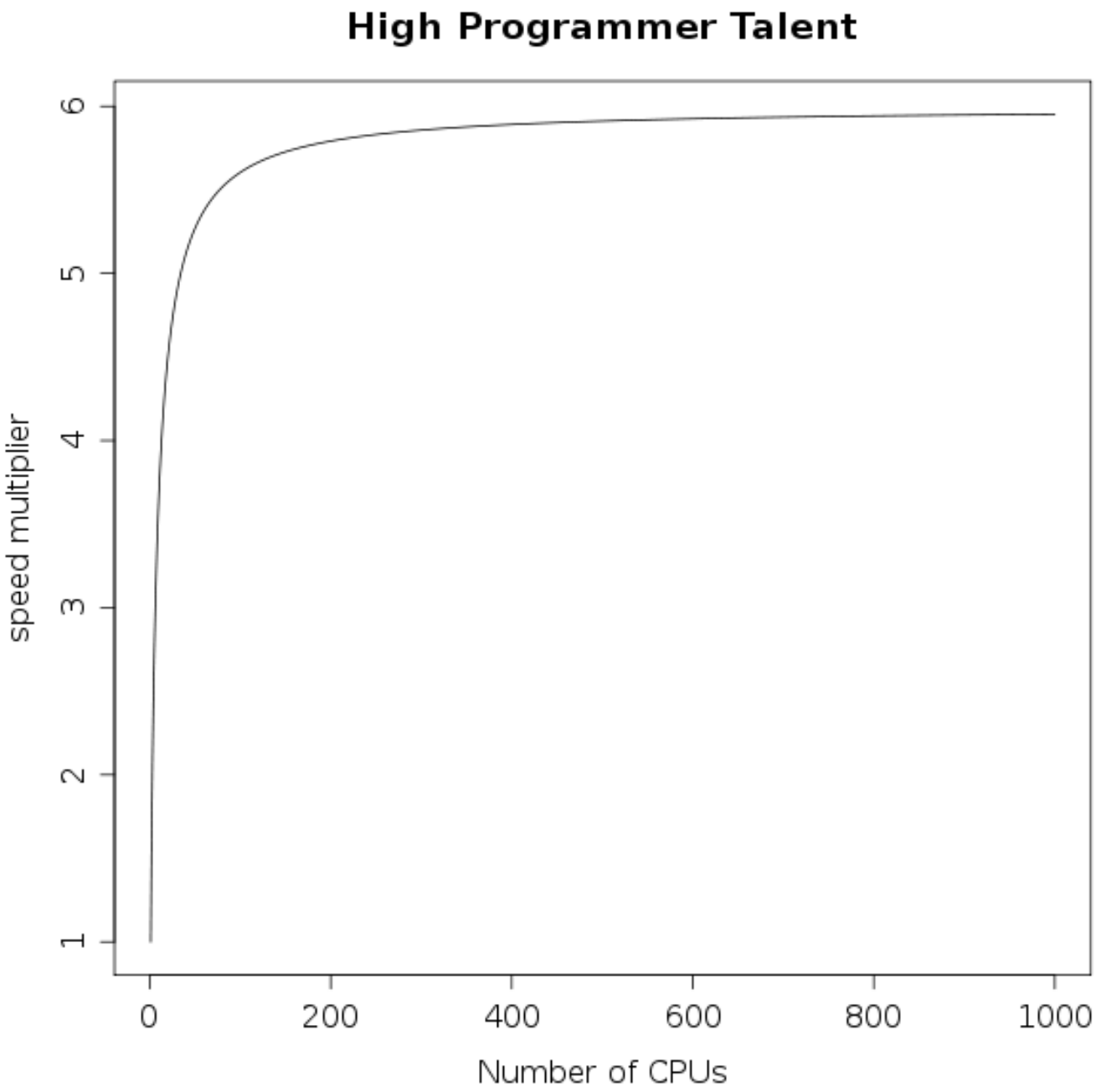
Amdahl's Law And Programmers

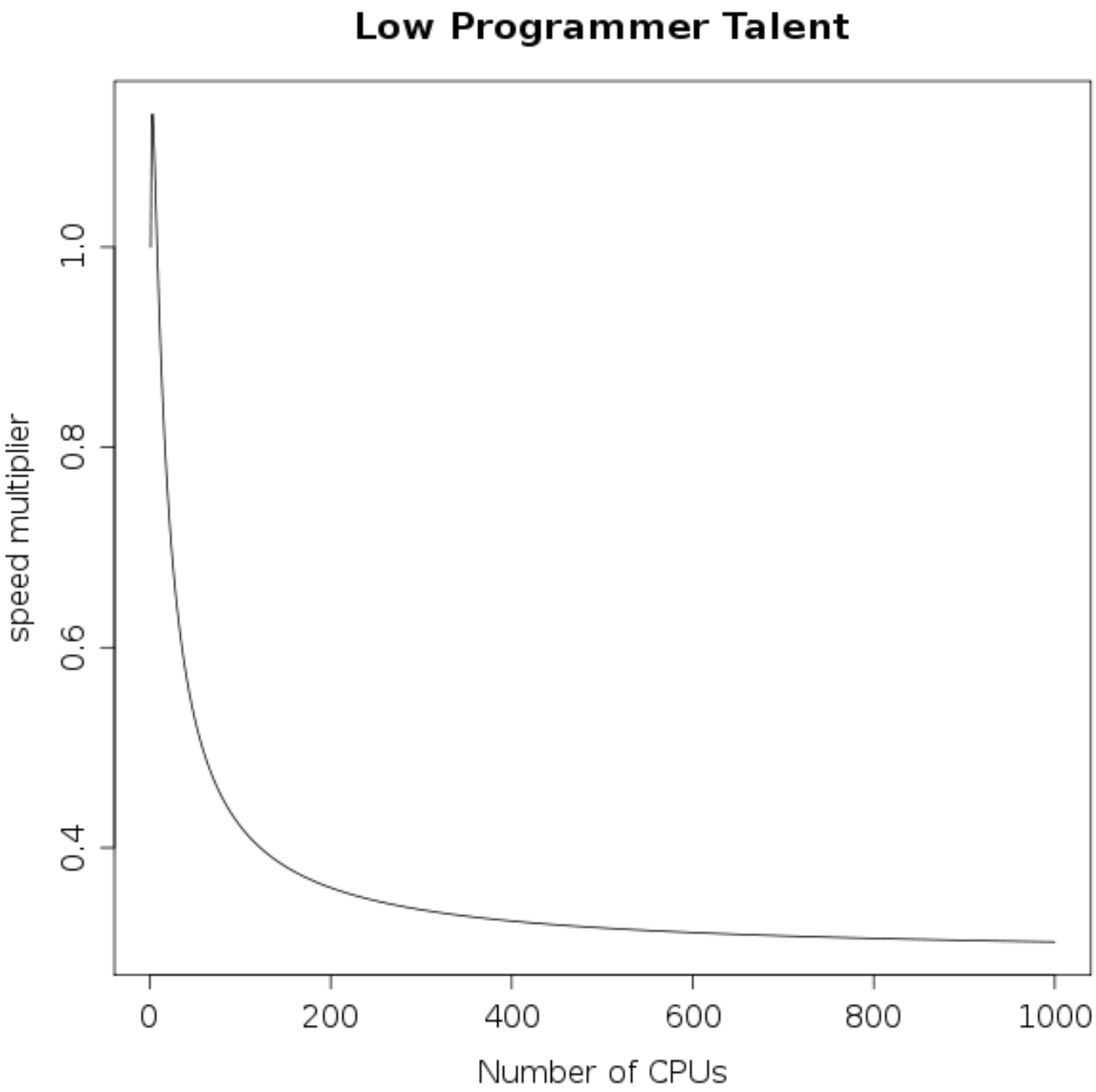
- Imagine that programmers aren't very good.
- I know, it's a stretch.
- Now, imagine that there's a range of values for f depending on how good they can parallelize.
- **f now becomes a normal curve with a mean and standard deviations determining the quality of parallel processing implementation.**

Amdahl's Law And Programmers

- Imagine that programmers aren't very good.
- I know, it's a stretch.
- Now, imagine that there's a range of values for f depending on how good they can parallelize.
- f now becomes a normal curve with a mean and standard deviations determining the quality of parallel processing **implementation**.
- **p becomes a fixed input of CPUs.**


Mean Programmer Talent
speed multiplier
1.0
1.5
2.0
2.5
3.0
0
200
400
600
800
1000
Number of CPUs


High Programmer Talent
speed multiplier
1
2
3
4
5
6
0
200
400
600
800
1000
Number of CPUs


Low Programmer Talent
speed multiplier
0.4
0.6
0.8
1.0
0
200
400
600
800
1000
Number of CPUs

Last Graph Was Bullshit

- **It is weirdly true, but that graph doesn't mean anything.**

The Message Is There

- **Throw as many cores as you want at the problem.**

The Message Is There

- Throw as many cores as you want at the problem.
- **The real limiter is programmer ability**

The Message Is There

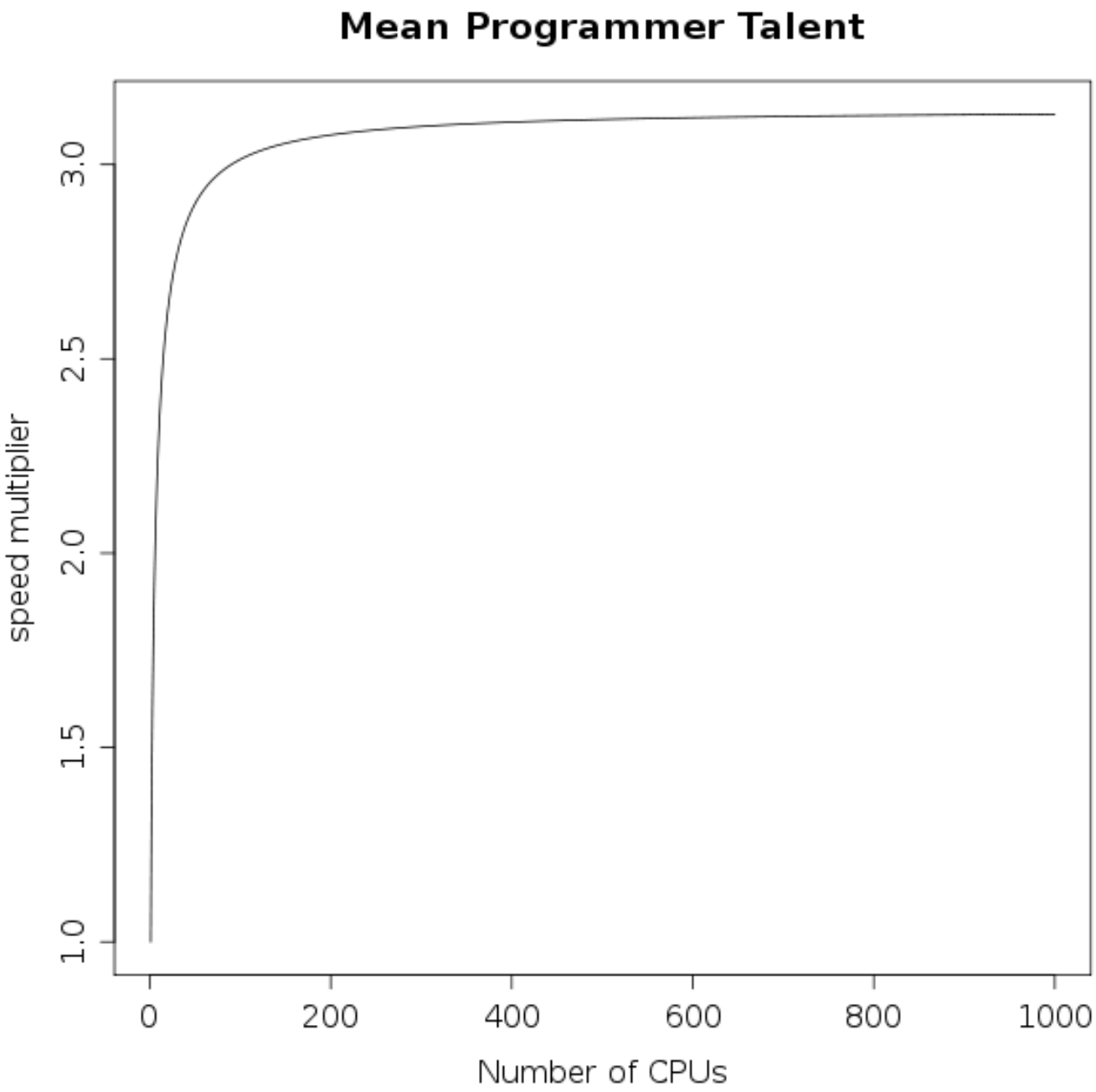
- Throw as many cores as you want at the problem.
- The real limiter is programmer ability
- **And a more complex Prolog like Erlang isn't helping.**

The Message Is There

- Throw as many cores as you want at the problem.
- The real limiter is programmer ability
- And a more complex Prolog like Erlang isn't helping.
- **Instead, the platform should make this easier.**

It's A Multiplier

- **Let's look at that graph again though.**


Mean Programmer Talent
speed multiplier
1.0
1.5
2.0
2.5
3.0
0
200
400
600
800
1000
Number of CPUs

It's A Multiplier

- **One thing the pimps of parallel don't tell you**

It's A Multiplier

- One thing the pimps of parallel don't tell you
- **You can also make your code faster**

It's A Multiplier

- One thing the pimps of parallel don't tell you
- You can also make your code faster
- **By...**

It's A Multiplier

- One thing the pimps of parallel don't tell you
- You can also make your code faster
- By...
- **Just making it fucking faster.**

Imagine A World

- **Where you aren't working with a Virtual Machine**

Imagine A World

- Where you aren't working with a Virtual Machine
- **Instead, the OS and CPU supports dynamic languages easily**

Imagine A World

- Where you aren't working with a Virtual Machine
- Instead, the OS and CPU supports dynamic languages easily
- **Where you can use something like EaRing to sort-of-compile, sort-of-interpret**

Imagine A World

- Where you aren't working with a Virtual Machine
- Instead, the OS and CPU supports dynamic languages easily
- Where you can use something like EaRing to sort-of-compile, sort-of-interpret
- **I'm not saying EaRing is the answer.**

Imagine A World

- Where you aren't working with a Virtual Machine
- Instead, the OS and CPU supports dynamic languages easily
- Where you can use something like EaRing to sort-of-compile, sort-of-interpret
- I'm not saying EaRing is the answer.
- **But Amdahl's law says putting slow languages on more CPUs is only so helpful.**

Fibonacci

- **Notice I wrote a fast Fibonacci sequence.**

Fibonacci

- Notice I wrote a fast Fibonacci sequence.
- **It wasn't too hard, here it is again.**

```
function fibit(in : ulong) : ulong
    getarg.ul(R2, in)
    movi.ul(R1, 1)
    blti.ul(exit:, R2, 2)
    subi.ul(R2, R2, 1)
    movi.ul(R0, 1)
loop:
    subi.ul(R2, R2, 1)
    addr.ul(V0, R0, R1)
    movr.ul(R0, R1)
    addi.ul(R1, V0, 1)
    bnei.ul(loop:, R2, 0)
exit:
    movr.ul(RET, R1)
    ret
end
```

Fibonacci Should Be Banished

- **It is too easy to game a measurement like this.**

Fibonacci Should Be Banished

- It is too easy to game a measurement like this.
- **I just used a fast goto loop.**

Fibonacci Should Be Banished

- It is too easy to game a measurement like this.
- I just used a fast goto loop.
- **You can even do tricks like static calculation and tables.**

Fibonacci Should Be Banished

- It is too easy to game a measurement like this.
- I just used a fast goto loop.
- You can even do tricks like static calculation and tables.
- **It shows only a minor type of computation, one that's not particularly useful.**

In Conclusion – You're Screwed

- **Amdahl's Law Says So.**

In Conclusion – You're Screwed

- Amdahl's Law Says So.
- **6x with the best programmers on about 200 CPUs**

In Conclusion – You're Screwed

- Amdahl's Law Says So.
- 6x with the best programmers on about 200 CPUs
- **Sounds like a great way to sell CPUs**

In Conclusion – You're Screwed

- Amdahl's Law Says So.
- 6x with the best programmers on about 200 CPUs
- Sounds like a great way to sell CPUs
- **Parallel don't matter nearly as much as Sun wants you to think**

In Conclusion – You're Screwed

- Amdahl's Law Says So.
- 6x with the best programmers on about 200 CPUs
- Sounds like a great way to sell CPUs
- Parallel don't matter nearly as much as Sun wants you to think
- **They are just trying to route around Moore's Law**

In Conclusion – You're Screwed

- Amdahl's Law Says So.
- 6x with the best programmers on about 200 CPUs
- Sounds like a great way to sell CPUs
- Parallel don't matter nearly as much as Sun wants you to think
- They are just trying to route around Moore's Law
- **And blaming you for it**

In Conclusion – You're Screwed

- Amdahl's Law Says So.
- 6x with the best programmers on about 200 CPUs
- Sounds like a great way to sell CPUs
- Parallel don't matter nearly as much as Sun wants you to think
- They are just trying to route around Moore's Law
- And blaming you for it
- **But who built things this way in the first place?**

What We Need

- **We need CPUs that run modern languages**

What We Need

- We need CPUs that run modern languages
- **Operating systems that run modern CPUs**

What We Need

- We need CPUs that run modern languages
- Operating systems that run modern CPUs
- **And languages that aren't fucking interpreted**

What We Need

- We need CPUs that run modern languages
- Operating systems that run modern CPUs
- And languages that aren't fucking interpreted
- **The way Ruby is.**

And...

- **EaRing Is Going To Change The World.**

And...

- EaRing Is Going To Change The World.
- **One insult at a time.**

Tools Used

- **GNU Lightning**

Tools Used

- GNU Lightning
- **Ragel Super Lexer**

Tools Used

- GNU Lightning
- Ragel Super Lexer
- **Lemon Parser Generator**

Tools Used

- GNU Lightning
- Ragel Super Lexer
- Lemon Parser Generator
- **Valgrind**

Tools Used

- GNU Lightning
- Ragel Super Lexer
- Lemon Parser Generator
- Valgrind
- **Vim**

Getting The Presentation

- **I'll have the whole Bazaar repository online later today.**

Getting The Presentation

- I'll have the whole Bazaar repository online later today.
- **bzr pull http://zedshaw.com/repository/rubyenrails2008**

Getting The Presentation

- I'll have the whole Bazaar repository online later today.
- bzr pull http://zedshaw.com/repository/rubyenrails2008
- **See all my mistakes and get all the code.**

Getting The Presentation

- I'll have the whole Bazaar repository online later today.
- bzr pull http://zedshaw.com/repository/rubyenrails2008
- See all my mistakes and get all the code.
- **Vellum: http://zedshaw.com/projects/vellum/**

Getting The Presentation

- I'll have the whole Bazaar repository online later today.
- bzr pull http://zedshaw.com/repository/rubyenrails2008
- See all my mistakes and get all the code.
- Vellum: http://zedshaw.com/projects/vellum/
- **EaRing: bzr pull http://zedshaw.com/repository/earing**

Questions?

- **About Anything**