Why Your Team Has Slowed Down, Why That's Worse than You Think

(And How to Fix It)

Edmund Jorgensen
@tomheon

Hut 8 Labs

QCon New York, 11 June 2014

“Latency”

Going to talk a lot about “Latency” of your dev team, by which I mean: how much calendar time passes between the request for something (feature, bugfix) and the safe, sane delivery of that thing.

A Tale in 3 Acts

1. **Why** to Invest in Latency Reduction
2. **How** to Invest in Latency Reduction
3. How to Get the Very Important People **On Board** with Your Investments

Act 1:
Why to Invest in Latency Reduction

One answer:
Deliver features and bugfixes
to your customers quicker
because they love that!

True, but if we only consider that reason, we'll under-invest in latency reduction, b/c the reality is even more economically powerful.

Thought Experiment:
A Tale of Two Dev Teams

Meet Michael.

He's a CEO.

This is Michael. Michael is the CEO of a company makes its money off big, enterprise contracts, and a prospect he's been after for a long time is entertaining proofs of concept on a project.

If he lands this contract, it could set his business up for 5 years. If he loses it, the company may go under. The prospect has submitted a call for proofs of concept and is going to look at them all–Michael's and the competition's–in 90 days, not earlier, so the "deliver early to customers" advantage is gone.

Two Options

Regular Dev Team

- Good proof of concept
- $100,000
- **90 days**

Insanely Fast Dev Team

- Good proof of concept
- $100,000
- **1 Second**

Michael has 2 options for dev teams to create the POC. One is a good, solid team that could complete the task in 90 days, for a total cost of $100,000. The other is an insanely fast team who charge $100,000 a **second**, but can complete the same POC in only one second of work.

To call this out more explicitly: both teams would produce the same quality work, at the same cost. **Only their latency differs.**

Does this matter, in a world where early delivery is disallowed?

Regular Dev Team =
No Choices to Make

If Michael chooses the regular dev team, he has no decisions to make about when they do their work. They have to start work on the POC today just to finish in time for presentations.

Insanely Fast Dev Team =
$\sim$7,776,000 Choices to Make

With the insanely fast team, on the other hand, he has almost 8 million choices to make (that being the number of seconds in 90 days), since he could build the POC in any one of them. But are any of these choices valuable?


120
90
140
100
160
110
180
120
km/h

One option that jumps out is having the Insanely Fast Dev Team produce the POC in the ver first second. Michael can't deliver it early by the prospect's rules, but are there any other options available to him?

Iterate

He can show the POC around to some decent proxies for the customer, get their reactions, and incorporate those reactions into a second version of the POC (which will cost him an additional $100,000–more on that later).

Abandon

Maybe when Michael presents the early POC to one of the customer proxies, he gets the following reaction:

"This looks great–all you need to do now is attach DNA samples and medical histories for each of these records and you've got something really valuable!"

Michael can abandon the project immediately and turn his money and attention to other projects that won't require **illegal data** to be valuable. He is 90 days ahead of his competition getting into other markets, while he knows that they're drawing dead with their own POCs.


120
90
140
100
160
110
180
120
km/h

So yeah, producing the POC in the first second can give Michael some huge advantages, even if he can't deliver early. That Insanely Fast Dev Team is looking pretty good right about now. But wait...there's more!


PROCRASTINATE
PROCRASTINATE

What if Michael goes to the other extreme, and waits until the last second to produce the POC–procratinates like there's no tomorrow–does that give him any interesting advantages?


TALK IS
CHEAP –
IT'S FREE!

Michael can use that time to do a ton of research (which is usually cheaper than development) before he commits to building. He can literally wait until the last second and use something he learned in the hallway waiting to present in order to produce a better POC.


PROCRASTINATE
PROCRASTINATE

So yes, there are benefits to waiting until the last minute.

Mix and Match!

But the real power of the Insanely Fast Dev team becomes apparent when Michael combines the "build early" and "build late" strategies. He does cheap research until he feels he'd gain more information with something to show. Now he builds (in a second) a POC, and gets feedback on it, rinsing and repeating until:

- He feels that a new iteration isn't worth $100,000
- He decides to scrap the project

Finish early, start late = Information Gain

What the Insanely Fast Dev Team gives Michael is the ability to (in the word of Don Reinertsen) **finish early** and **start late**, which allows for information gain: tasks that finish early tend to generate information, and tasks that start late can benefit from newly available information.

In the presence of uncertainty–which software development is all about–information is value. So this ability to finish early and start late, thereby gaining information, is tremendously valuable.

The poor old regular dev team, with their higher latency, has to start early and finish late just to get the job done. They can neither generate information for later tasks, or take advantage of newly available information.

Which team

should Michael choose?

Shut up and take my money!

It's a formality at this point–he should choose the Insanely Fast Team and their reduced latency–in fact, he should be willing to pay much more than $100,000 to get access to that team.

Michael is psyched to have the option to
spend more on
invest more with
the Insanely Fast Development Team.

Sinking an extra million, even, into a much improved chance to land a 25 million dollar contract is an **investment**, not a "cost" (the difference being: you expect an investment to return to you, increased).

(Footnote: Realistic Drops in Latency)

Quarterly Release to Hourly Release

90 Days * 24 Hours =

$\sim$**2,000x** reduction in latency

Easy objection: sure, going from 90 days to 1 second would be awesome, but in the real world that's impossible. But it's not impossible to reduce latency on simple features, bugfixes, etc. from quarterly to hourly releases (it requires real investment, but many places have done it). That's huge.

Why to Invest in Latency Reduction

- Reducing latency **creates** information
- Information == (probabilistic) $
- Reducing latency **creates** (probabilistic) $

Act 2:
How to Invest in Latency Reduction

Even once we understand **why** to invest in latency reduction, it's not always easy to actually do it.

Some Common Lat[illegible]uction Strategies

- H[illegible]eHus[illegible]
- W[illegible]rk EthicW[illegible]Ethic
- [illegible]DGSD

I don't want to talk about these.


hustle
grind
work ethic

Like, at all.


ONE DOLLAR
E 95074985 D
THE UNITED STATES OF AMERICA
$100

2 things: these “solutions” are cop-outs–trying to turn an organizational and economic concern into something moral and personal. Also, as “solutions” they don’t work particularly well.

"There's nothing else to do today–let's reduce some latency."

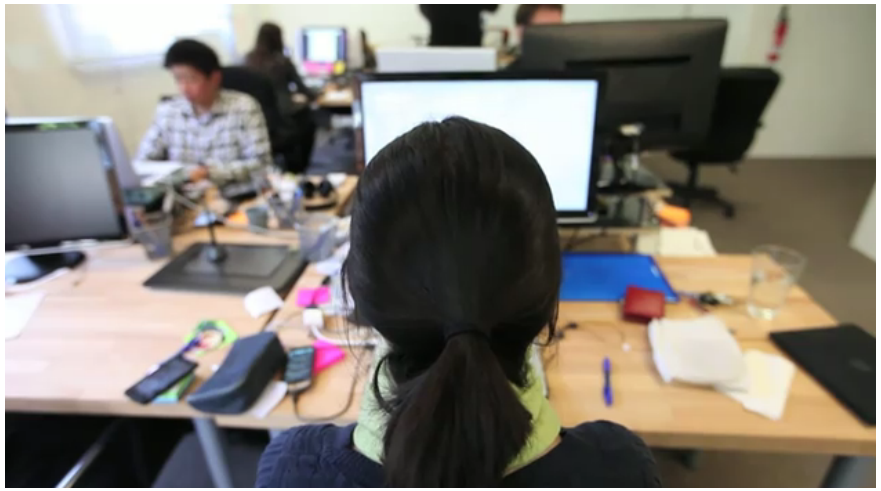
a) Abraham Lincoln
b) Larry Page
c) **No one, ever**

First key for how to invest in reducing latency is to realize that it never happens in a vacuum, but in the day to day insanity of a real business, where it's hard to see latency.

Case Study: A Tale of Two Engineers

Let's get at this through a case study.

Meet Cindy.



She's an engineering leader.

Cindy is an engineering leader at a company building social networks for animals. They're killing it. When she arrives one morning two of her engineers are waiting for her.

Bruce: SO MANY MIGRAAAAAATIONNNNNNS

Bruce complains that he's spending half an hour every morning getting production db migrations live for people, since he's the only one who understands the production db enough to do it. He wants to stop working on features for a week to write a migration script.

Roy: FROBULATOR RAGE...BUILDING

Roy is angry about the quality of the code in the Frobulator service, which he recently rediscovered while updating copyright headers. It was banged out crappily in PHP and now he wants to stop working on features to rewrite it in Scala.


NO
SEE YOUR ANALYST
SIT ON IT
YES
FIRE SOMEONE
PASS THE BUCK
MAYBE
REORGANIZE!

So what would we do, if we were in Cindy's shoes? We can't make everyone happy–either features are going to get pushed back, or these engineers' projects are.

Your time is valuable, Bruce.

One way to handle it: Bruce, your time is valuable, **because** of your expertise with the db. We have to invest it wisely. If a 40 hour investment in a migration script saves you 1/2 a day, that's 80 business days to recoup invsestment–too long!

Think of the customer, Roy.

We explain to Roy: our customers can't tell if the Frobulator service is bad PHP or transcendant Scala, and they don't care. Being focused on the customers and the business means we have to be super jealous of hours we spend on projects that are invisible to customers.


ONE DOLLAR
E 95074985 D
THE UNITED STATES OF AMERICA
$100

So we feel good–we've taken an economic approach, stayed focused on the customer, and saved our business a pile of money. Or maybe...

We have just lit a bunch of money on fire. What's going on here? How did we cost our business so much money when we thought we were saving it?

We have fallen for two of the classic blunders.

Classic Blunder #1: Obsessing over "Paid Engineer Hours" instead of Latency

How we experience the world:

- Paid engineer hours = expensive
- Latency = (merely) annoying

How the world really is:

- Paid engineer hours = expensive
- Latency = **MASSIVELY** expensive

Classic Blunder #2: Morality

Michael the CEO got to write a check to reduce latency...

Which feels clean and easy, without the moral or psychological barriers.

...we often have to invest engineer hours.


2
42
innovation

Which comes with a lot of baggage. It feels a litttle **wrong** for engineers to spend paid hours making their own lives better.

Won't somebody please think of the customers?

It feels like, morally, they should be spending that time on the customers. I mean, aren't jobs **supposed** to suck?

But if we love the customer, we want to stay in business, and be able to deliver in a timely fashion indefinitely. That's what latency reduction is about.

A Tale of Two Engineers (Redux)

With these blunders in mind, let's revisit Bruce and Roy.

WAAH I'M TIRED OF MY JOB AND NEED TO BE CAT HERDED

This is basically what we heard from Bruce, if we're being honest about our reaction.

Danger, Will Robinson!

This is what we **should** have heard from Bruce–reality sending us an early warning. “Latency is brewing in your engineering org!”

What happens if Bruce takes a...gasp...vacation? The queue will grow while he's gone, and nothing will go out. What if Bruce silently decides to be more "efficient" and only push migrations once a week? Our latency has just skyrocketed, catastrophically and invisibly.

We should let Bruce write his migration script–this is a no brainer.

WAAH I'M A PERFECTIONIST AND NEED TO BE CAT HERDED

This is basically what we heard from Roy.


A CLUE!!!

This is what we should have heard.

Roy told us he was in the Frobulator service to “update copyright headers.” Is this a clue that he hasn’t been in the Frobulator service code for at least a year? Why not? Is this the case with other engineers too?

If the Frobulator Service is just frobulating fine, and needs no work, great.

If, on the other hand, the engineers are terrified of it because it's such a mess, and don't make changes there, it could be introducing latency.


You
Latency

There are no top down solutions to latency. Latency will always creep in. A new QC process here, another step there... We should be calmly, bravely terrified, and always on the lookout, remembering that our “gut reactions” about the expense of latency are probably wrong.


KEEP
CALM
AND
CONSTANT
VIGILANCE

And we should teach this calm, brave terror to our whole team. No one person can see all the brewing latency. Make it something everyone hates, seeks out, and termintes.

Good Bets for Getting Started in Latency Reduction Investment

Generally:
Operational/Processy beats
Frameworky
(Especially at the beginning)

Please don't make me spin up a new server...

Find things that engineers complain about–latency is often experienced as annoying–and invest in making them better.

"Only Joe can make changes to the user service."

Just like with Bruce, having one person be the “super hero” of some piece of code / process creates latency. Make them the super hero for training others instead.

Tests!

Good tests reduce latency because they allow you to move faster and spend less time fighting fires.

I AM IN UR SERVERS

MONITORING UR VALUE CREATION

Good monitors reduce latency because they allow you to move faster and spend less time fighting fires.

Keep it Incremental FTW

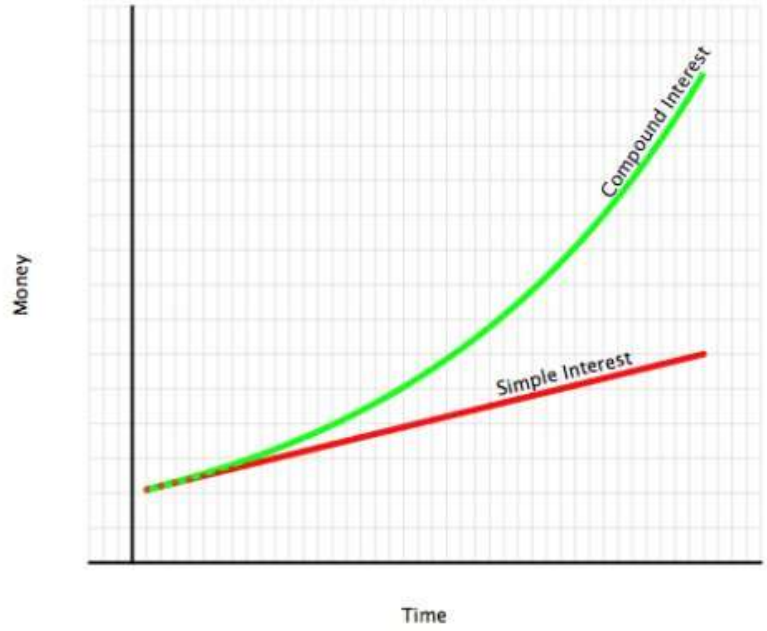
But when it's finished...

...it will be AMAZING.

This is how many / most engineers want to handle these kinds of projects–tear everything down and build it right! Terrible idea in general, and especially with latency reduction.

Because latency reduction has this extra weird recursive nature, in that investments in latency reduction **speed up the same channels** by which we make future latency reduction.

“The most powerful force in the universe.”



–Albert Einstein (maybe)

That makes latency reduction a form of **compound interest**. And just like with your retirement account, small frequent investments earn more compound interest overall compared to big infrequent investments.

Recap: How to Invest in Latency Reduction

- Beware the Classic Blunders ("paid engineer hours" vs. latency, morality)
- Cultivate a Calm Team-Wide Terror of Latency
- Start with Operational and Process Wins
- Keep it incremental

Act 3:

How to Get the Very Important People **On Board** with Your Investments

These are the bosses, bosses' bosses, and so on. They sign the checks and make a lot of the decisions. How do we get them on board with our latency reduction investments?

Allegory of the Warehouse

Meet Bob.

He's a foreman in a warehouse.

Bob has a problem.

Latency is way up in the warehouse, which means profits are down and the VIPs are not happy. The latency is being caused by a problem that a lot of warehouses have these days...

Ninjas. Obstructionist ninjas who pop out in the aisles and...

Insist that forklift operators juggle to entertain them before being allowed to be on their way.

Ninja Convention...

Sold out!

Even worse, the ninjas sense when VIPs are near and, being jerks, melt into the shadows, so the VIP sees only a forklift operator juggling.

Allegorical Key:

Bob = engineering leader
Forklift operators = engineers
Ninjas = sources of latency

Solutions to Bob's / our problem?

Hire more!

This is a natural response, but usually doesn't turn out well–we're adding bandwidth, not reducing latency. One ninja can block 4 operators in an aisle–one bottleneck getting changes to production can hold back 10 engineers.

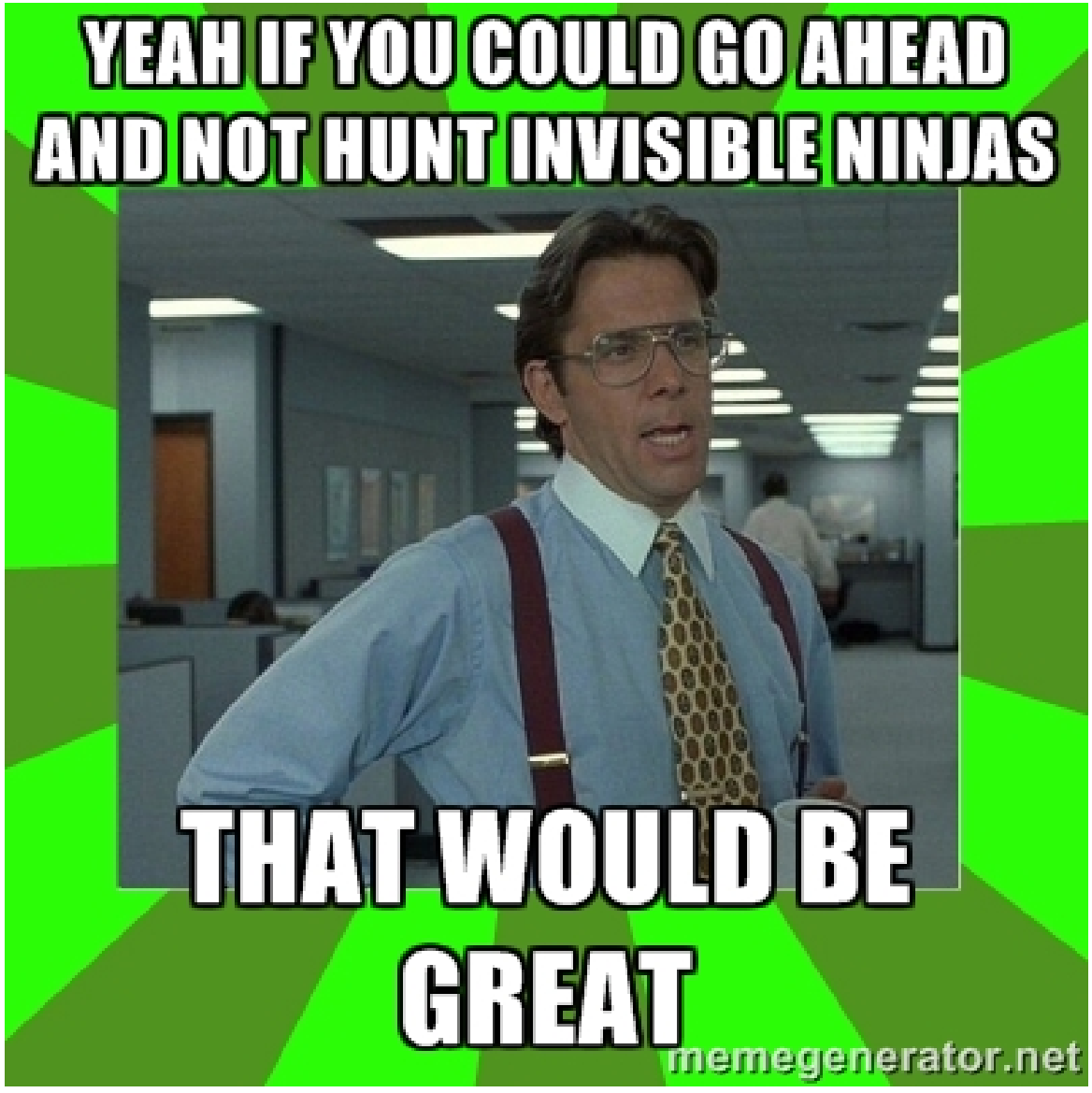
Hire different!

Maybe Bob hires only professional jugglers, who can please the ninjas quicker. Maybe we hire only those mythical 1% of engineers who can juggle all the problems and still move fast. This is better than just hiring, but isn't scaleable.

Bring the fight to the ninjas.

Maybe Bob's team spends Thursday afternoons hunting down the ninjas. They'll come back, but until then there's less latency. Maybe we, as engineers, spend some of our time on reducing latency.

Only one small problem...


YEAH IF YOU COULD GO AHEAD AND NOT HUNT INVISIBLE NINJAS
THAT WOULD BE GREAT
memegenerator.net

Often to the VIPs at software businesses, the causes of latency are just as invisible as these ninjas. We sound crazy if we talk about them.

“You’re telling me you want to forklift stuff faster... by taking butts out of forklifts?”

s/butts in forklifts/fingers on keyboards/

The first rule of selling ninja hunts to the Very Important People is...

Do not talk about ninjas.

The second rule of selling ninja hunts to the Very Important People is...

DO NOT TALK about ninjas.

Ninja Hunt Taboo–the Game

- Database
- Optimize
- Technical Debt
- etc.
- etc.
- etc.

Treat selling latency reduction like a game of Taboo. Don't mention anything that sounds like the ninjas.

Don’t be this guy...

Be **this** guy.

Because inside every Very Imporant Person...

Is this guy.

Inside the VIP is an inner child who just wants to go fast–make decisions and see the results right away.

The VIP Experience...

Latency is way up

Quick Wins are way down

But this has been the experience since the VIP joined your business.


SLANDERS
56
1

So how do we sell latency reduction to them? By selling “speeding up” to that inner child. And we make the first one free...


ADOPT
A
HIGHWAY
WARD'S HOLLOW

Choose a VIP-visible thing that's slow in your business–e.g., how long it takes a new advertiser to get on the site. Do a skunkworks project (off the books!) to improve it. Then advertise your sponsorship. Repeat a few times.

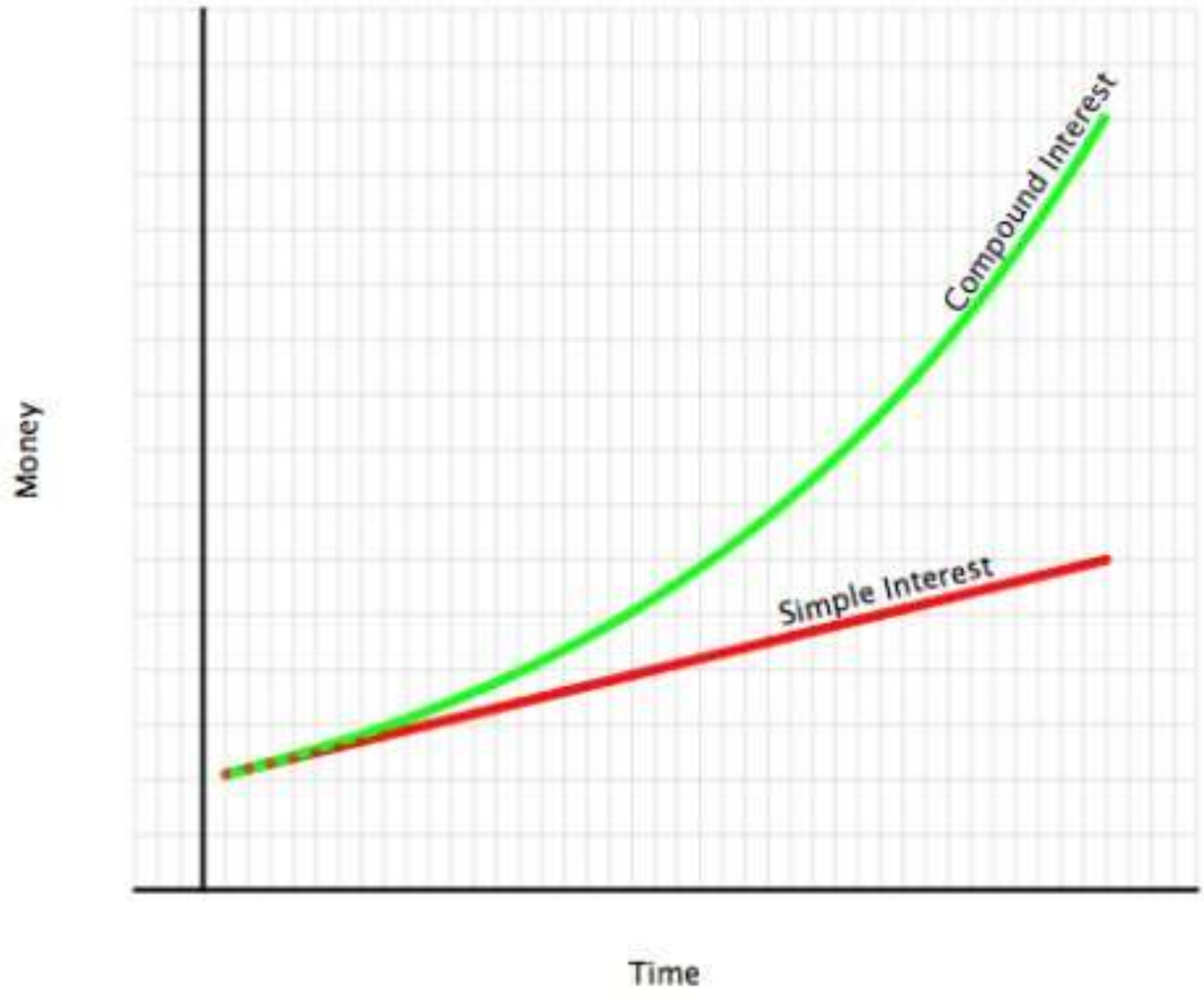
Aim for **30%**.

Use the generated good will to carve out a constant percentage of time for speeding up. Aim for 30%. It sounds too high. It isn't.

Let's work on features instead...

Just this once.

You want a percentage, not to have to argue for projects piecemeal, because features will **always** feel more important. Tie yourself to the mast with a constant percent of time.


Money
Compound Interest
Simple Interest
Time

Also because a constant percent investment is a great way to invest in compound-interest bearing accounts–like latency reduction.

ABAAH

(Always be Adopting a Highway)

Never stop sponsoring the speed up of those highly visible slow things in the business, because that percentage time is always going to be at risk.

Recap and Parting Exhortation

1. Sell speed to the VIP's inner child with some visible, skunkworks wins
2. Use generated goodwill to carve out a constant percentage of time for latency reduction
3. Keep promoting the latency reduction wins
4. Profit!


DANKE MERCI THANK YOU GRACIAS ARIGATO
DANKE MERCI THANK YOU GRACIAS ARIGATO
DANKE MERCI THANK YOU GRACIAS ARIGATO
DANKE MERCI THANK YOU GRACIAS ARIGATO
DANKE MERCI THANK YOU GRACIAS ARIGATO
DANKE MERCI THANK YOU GRACIAS ARIGATO
DANKE MERCI THANK YOU GRACIAS ARIGATO
DANKE MERCI THANK YOU GRACIAS ARIGATO
DANKE MERCI THANK YOU GRACIAS ARIGATO
DANKE MERCI THANK YOU GRACIAS ARIGATO
DANKE MERCI THANK YOU GRACIAS ARIGATO
DANKE MERCI THANK YOU GRACIAS ARIGATO
DANKE MERCI THANK YOU GRACIAS ARIGATO
DANKE MERCI THANK YOU GRACIAS ARIGATO
DANKE MERCI THANK YOU GRACIAS ARIGATO
DANKE MERCI THANK YOU GRACIAS ARIGATO
DANKE MERCI THANK YOU GRACIAS ARIGATO
DANKE MERCI THANK YOU GRACIAS ARIGATO
DANKE MERCI THANK YOU GRACIAS ARIGATO
DANKE MERCI THANK YOU GRACIAS ARIGATO
TO AVOID DANGER OF SUFFOCATION, KEEP THIS BAG AWAY FROM BABIES AND CHILDREN. DO NOT USE IN CRIBS, BEDS, CARRIAGES OR PLAYPENS. THIS BAG IS NOT A

More Reading

- *The Principles of Product Development Flow*, Donald Reinertsen
- "How to Survive a Ground Up Rewrite", Dan Milstein
- "Speeding Up Your Engineering Org, Part I (Beyond the Cost Center Mentality)", Edmund Jorgensen
- *The Goal*, Eliyahu M Goldratt

Image Credits

- Piggy Bank image courtesy of http://401kcalculator.org
- Michael image from https://www.flickr.com/photos/tristanreville/11297541166
- Stack of chips from https://www.flickr.com/photos/adrian_s/17623439
- Speedometer image from https://www.flickr.com/photos/thatguyfromcchs08/2300190277
- Young engineer from https://www.flickr.com/photos/mightyohm/2645244056
- Boat from https://www.flickr.com/photos/lukaszduleba/8333830430
- Procrastinate pennants from https://www.flickr.com/photos/oliviaew/8387008029
- Talk is Cheap from https://www.flickr.com/photos/garryknight/6748884071
- Mismatched socks from https://www.flickr.com/photos/_nezemnaya_/3008150036

Image Credits cont.

- Mushroom Cloud image from https://www.flickr.com/photos/epicfireworks/2861531483/
- Bruce image from https://www.flickr.com/photos/mike9alive/2745045822
- Burning money from https://www.flickr.com/photos/purpleslog/2924979423
- Baggage from https://www.flickr.com/photos/aresauburnphotos/5973161314
- Boy with dragon shadow from https://www.flickr.com/photos/cadencrawford/8422302030
- Sad dog from http://www.flickr.com/photos/90543828@N00/379384313/
- Super Hero Joe from https://www.flickr.com/photos/28208534@N07/2874639194/
- Land Kayak from https://www.flickr.com/photos/romeo66/2663946311
- Monitor lizard from https://www.flickr.com/photos/arnolouise/6121333054

Image Credits cont. cont.

- Construction pit from https://www.flickr.com/photos/kenyee/2864004641
- Ouroborus from https://www.flickr.com/photos/lwr/3066148864
- Forklift operator from https://www.flickr.com/photos/toyotamheurope/6890427337
- Ninja from https://www.flickr.com/photos/thekellyscope/5057877769
- Juggler from https://www.flickr.com/photos/helico/404640681
- Convention hall from https://www.flickr.com/photos/whinger/7689179230
- Forklift fleet from https://www.flickr.com/photos/39955793@N07/8620134562
- Fire juggler from https://www.flickr.com/photos/neeravbhatt/6263155845
- Lecture hall from https://www.flickr.com/photos/uniinnsbruck/3722413559

Image Credits cont. cont. cont.

- Adopt a highway from https://www.flickr.com/photos/amayzing/9469215713
- Waterhouse's Odysseus and Sirens from https://www.flickr.com/photos/ngmorieson/6081346983
- Thank you bag from https://www.flickr.com/photos/theredproject/3301279921
- Decision doodad from https://www.flickr.com/photos/garrettc/91385737