C++, Networking, and Numerics

From Incremental Statistics to Online Machine Learning

Matt P. Dziubinski

CppCon 2015

matt@math.aau.dk // @matt_dz
Department of Mathematical Sciences, Aalborg University
CREATES (Center for Research in Econometric Analysis of Time Series)

Goal(s)

```
do
{
```

- Get some numbers,
- crunch them,
- get some more

```
}
while (more_numbers);
```

Outline

- Takeaways

Outline

- Takeaways
  - What I set out to do

Outline

- Takeaways
  - What I set out to do
  - What I actually did

Outline

- Takeaways
  - What I set out to do
  - What I actually did
  - Why and how what I actually did was very, very wrong

Outline

- Takeaways
  - What I set out to do
  - What I actually did
  - Why and how what I actually did was very, very wrong
  - What I should have done instead

Outline

- Failures
  - What I set out to do
  - What I actually did
  - Why and how what I actually did was very, very wrong
  - What I should have done instead

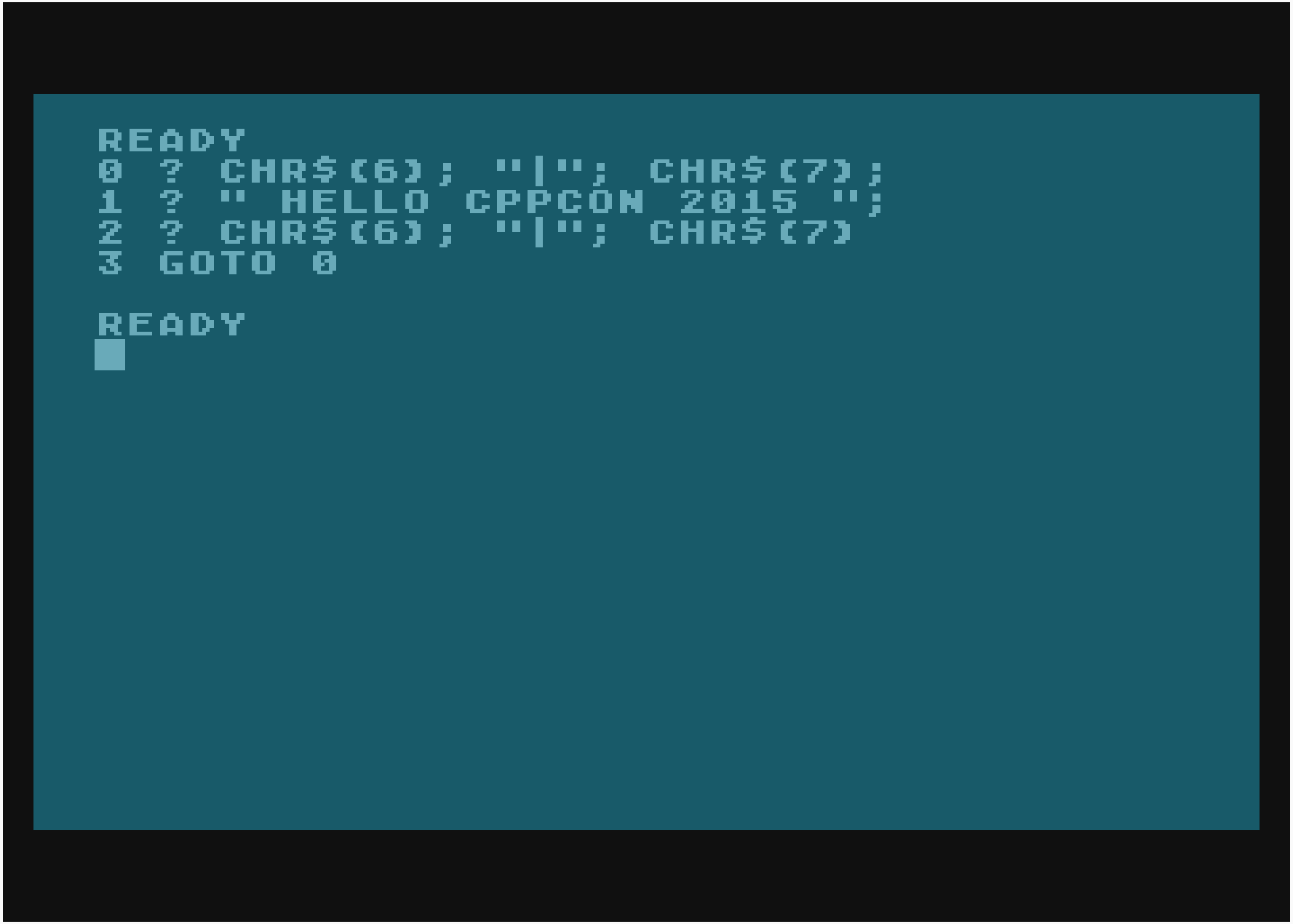
Outline

- Lessons learned
  - What I set out to do
  - What I actually did
  - Why and how what I actually did was very, very wrong
  - What I should have done instead

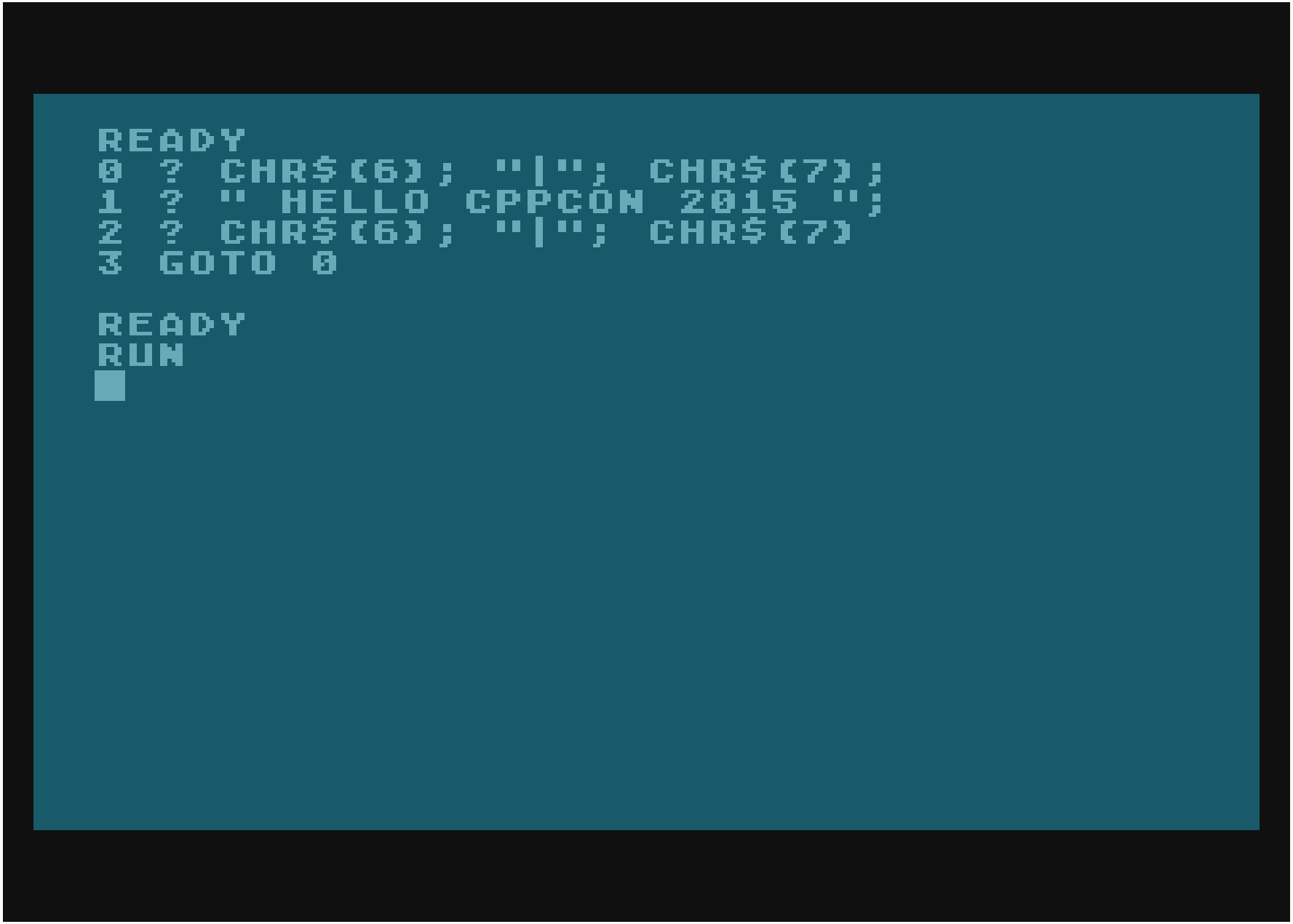
Intro

Intro

Intro


READY
0 ? CHR$(6); "|"; CHR$(7);
1 ? " HELLO CPPCON 2015 ";
2 ? CHR$(6); "|"; CHR$(7)
3 GOTO 0
READY

Intro


READY
0 ? CHR$(6); "|"; CHR$(7);
1 ? " HELLO CPPCON 2015 ";
2 ? CHR$(6); "|"; CHR$(7)
3 GOTO 0
READY
RUN

Intro

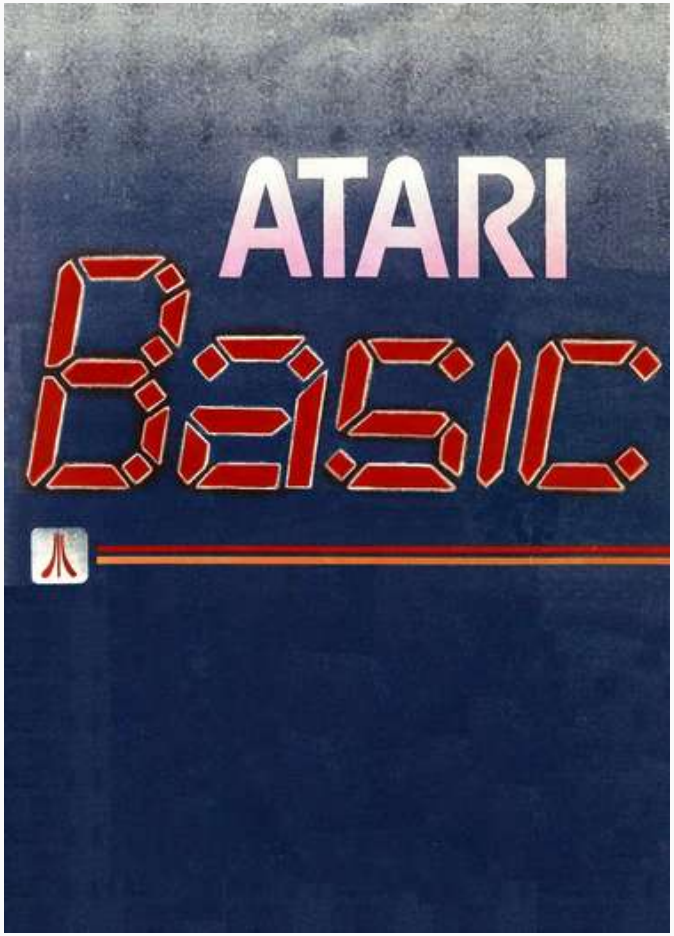

READY
0 ? CHR$(6); "|"; CHR$(7);
1 ? " HELLO CPPCON 2015 ";
2 ? CHR$(6); "|"; CHR$(7)
3 GOTO 0
READY
RUN
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015

Intro


HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HELLO CPPCON 2015
HEL

Intro: Disclaimer

- Atari BASIC - yay, my first programming language! :-)



Intro: EWD is not amused

Intro: oh...

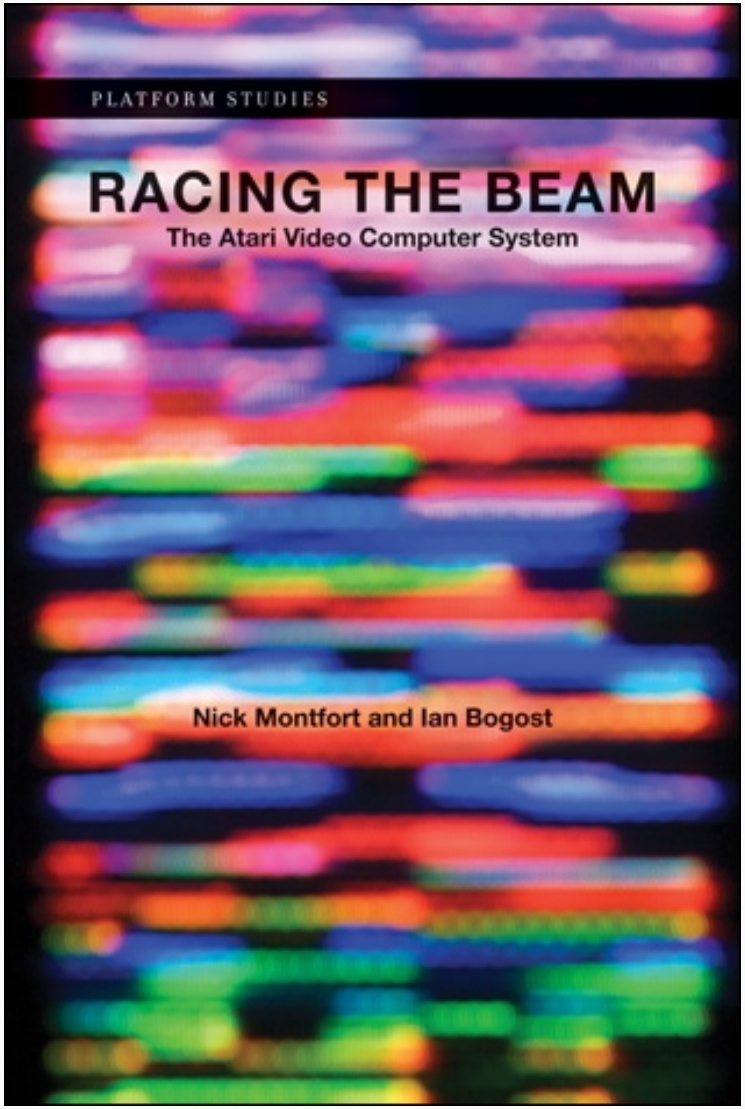
It is practically impossible to teach good programming to students that have had a prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration.

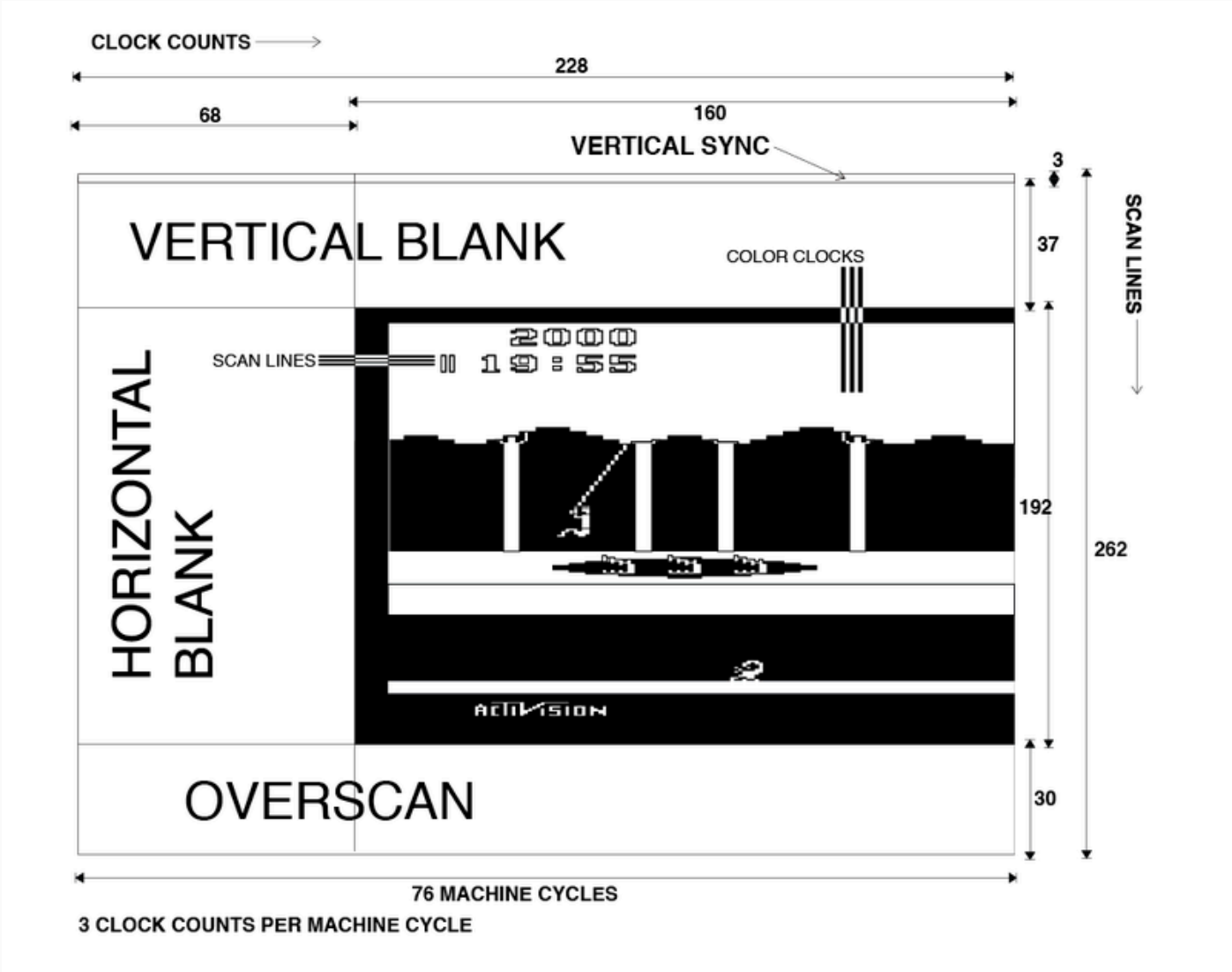
-- Edsger W.Dijkstra, ”How do we tell truths that might hurt?”, June 18, 1975

Is "My Bio" Slide Obligatory?

> It is practically impossible to teach good programming to students that have had a prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration.

-- Edsger W.Dijkstra, ”How do we tell truths that might hurt?”, June 18, 1975 *(emphasis mine)*


PLATFORM STUDIES
RACING THE BEAM
The Atari Video Computer System
Nick Montfort and Ian Bogost

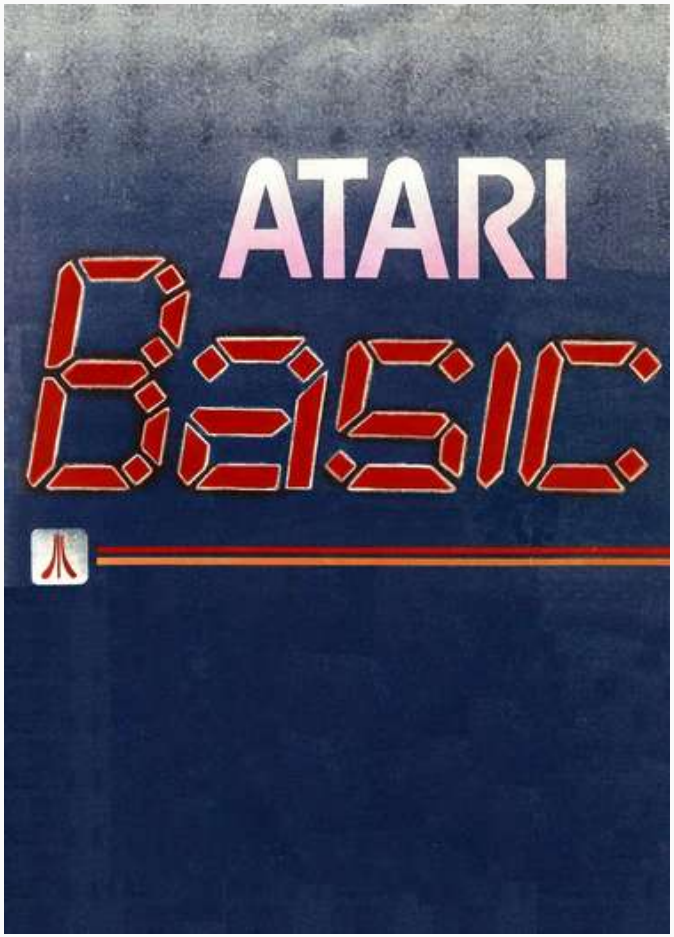

CLOCK COUNTS
228
68
160
VERTICAL SYNC
3
VERTICAL BLANK
COLOR CLOCKS
37
SCAN LINES
SCAN LINES
HORIZONTAL BLANK
192
262
ACTIVISION
OVERSCAN
30
76 MACHINE CYCLES
3 CLOCK COUNTS PER MACHINE CYCLE

Intro

Intro

```
READY
? PEEK(710)
148

READY
POKE 712, 148

READY
```

```
READY
POKE 712, PEEK(710)

READY
```

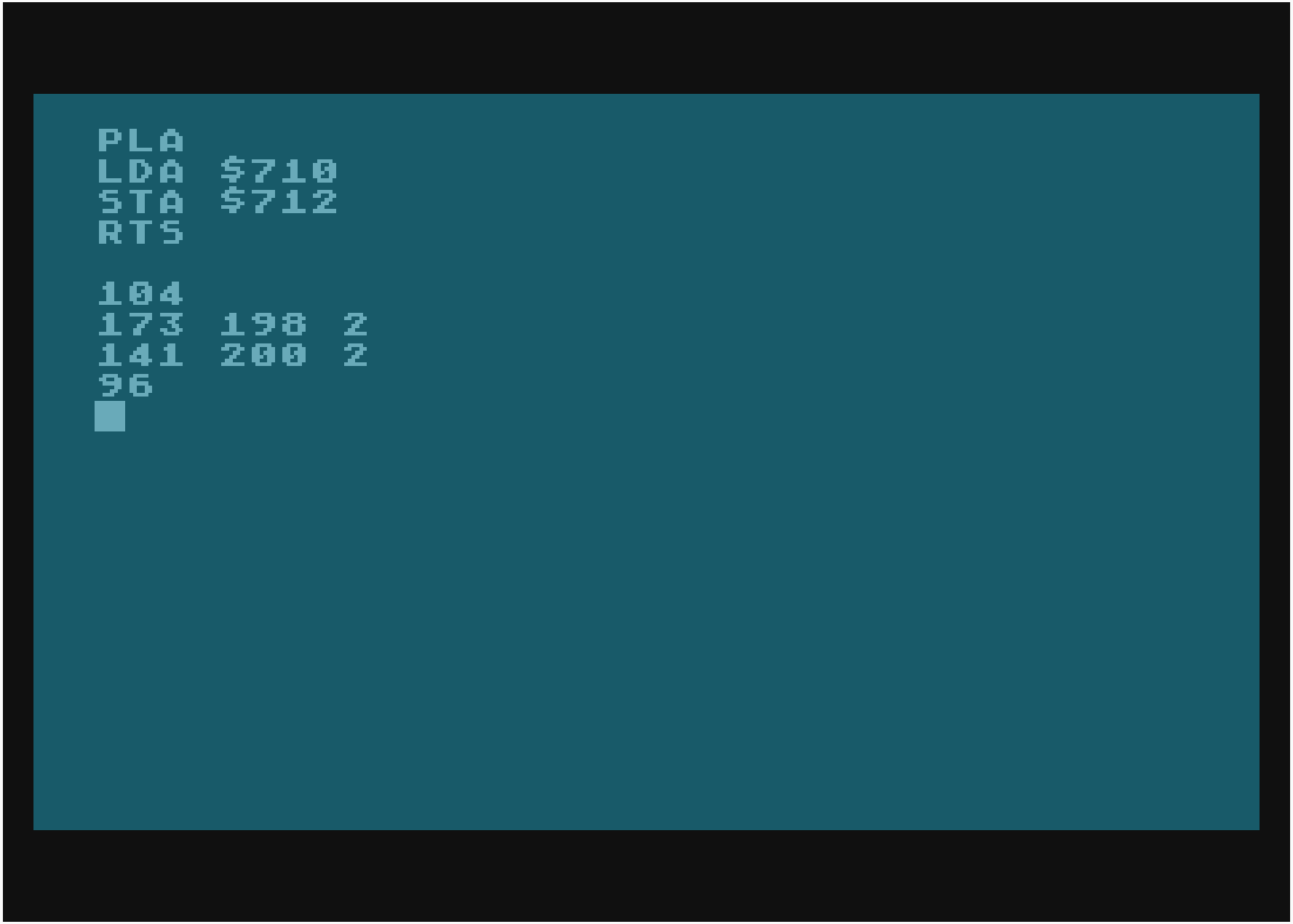

ATARI
BASIC

```
 5 FOR I = 1536 TO 1543
15 READ X
25 POKE I,X
35 NEXT I
45 DATA 104,173,198,2,141,200,2,96
```

Intro

Intro

Intro



```
READY
5 FOR I = 1536 TO 1543
15 READ X
25 POKE I, X
35 NEXT I
45 DATA 104, 173, 198, 2, 141, 200, 2,
 96
```

```
READY
5 FOR I = 1536 TO 1543
15 READ X
25 POKE I, X
35 NEXT I
45 DATA 104, 173, 198, 2, 141, 200, 2,
  96

V = USR(1536)

READY
```

Intro: Speedup

```
READY
POKE 540, 255:FOR I = 1 TO 10:POKE 712
, PEEK(710):NEXT I:TIME = PEEK(540):?
TIME:? 255 - TIME:? (255 - TIME)/60
250
5
0.0833333333

READY
POKE 540, 255:FOR I = 1 TO 10:V = USR(
1536):NEXT I:TIME = PEEK(540):? TIME:?
 255 - TIME:? (255 - TIME)/60
252
3
0.05

READY
? 5 / 3
1.66666666

READY
```

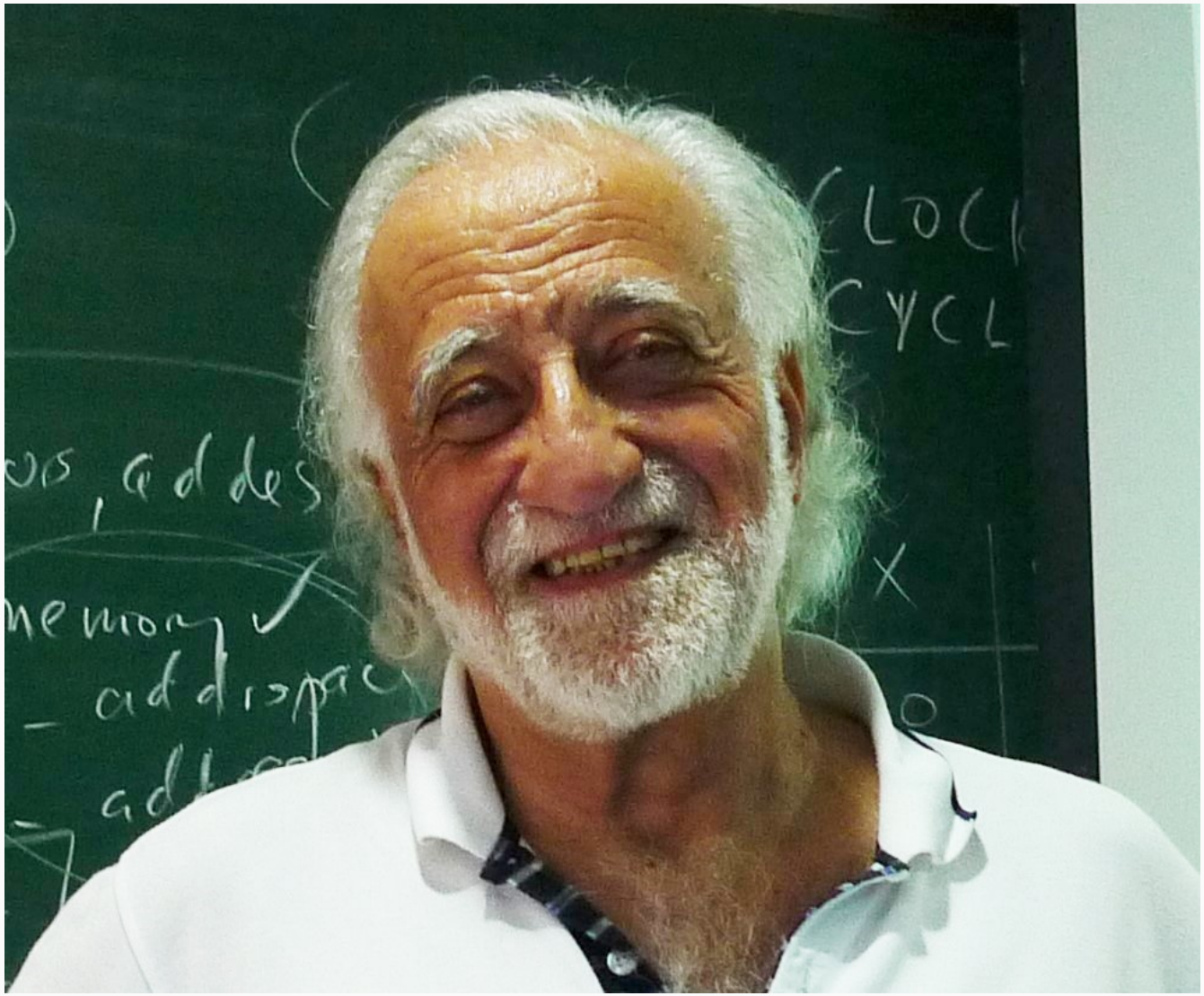
Intro: Takeaways

- Hardware matters for software

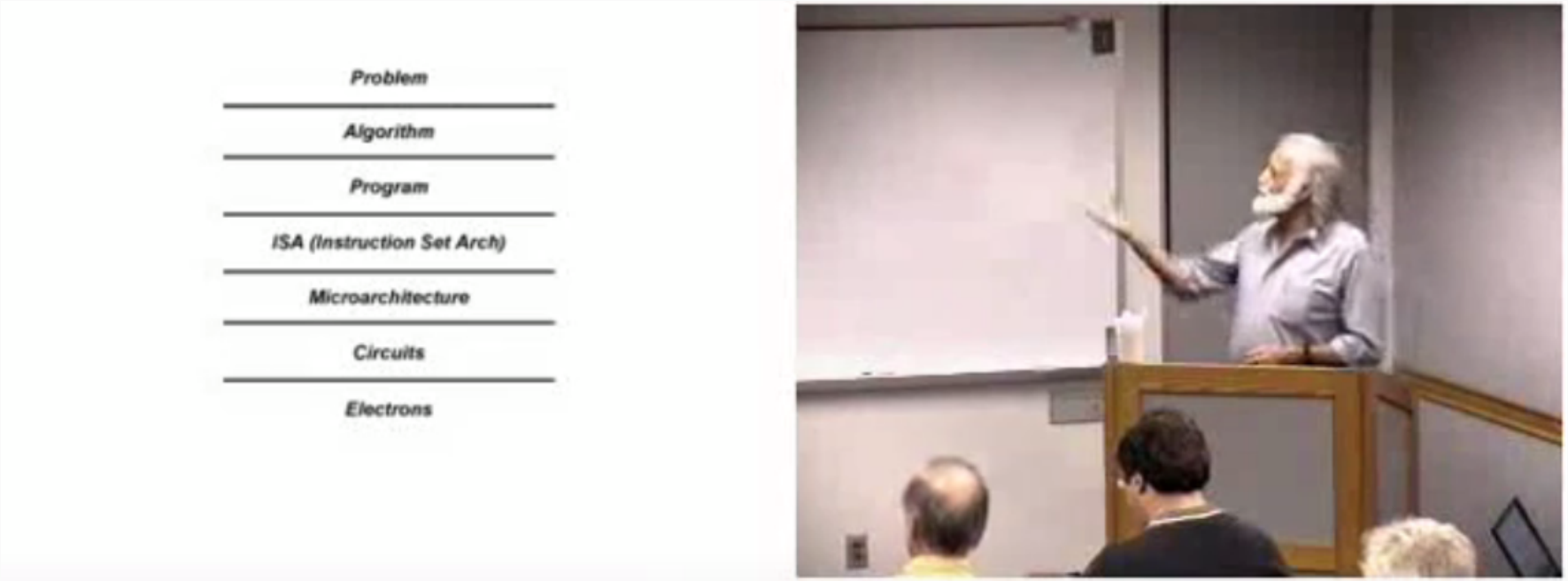
Intro: Takeaways

- Hardware matters for software
- Physics matters for hardware

Intro: Takeaways



Intro: Takeaways



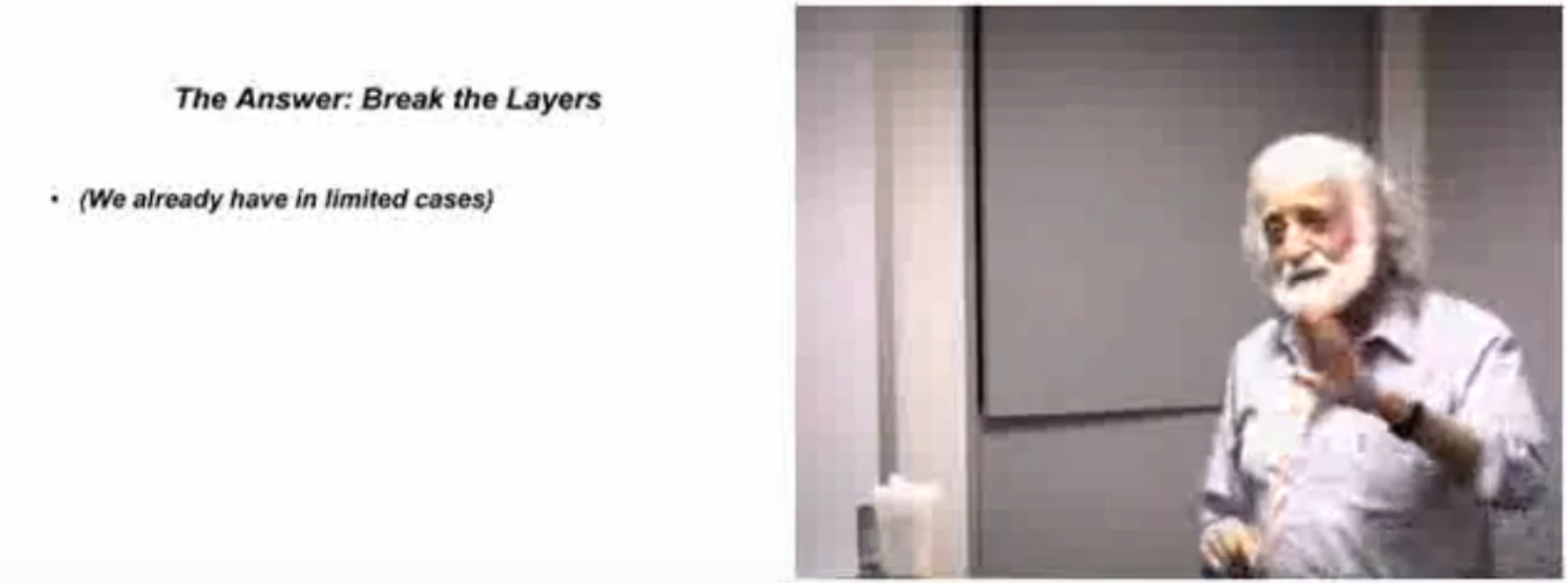
https://youtube.com/watch?v=0fLlDkC625Q

Intro: Takeaways

Until Recently (Phase I)

- *Maintain the artificial walls between layers*
- *Keeps the abstraction layers secure*
  - *Makes for a better comfort zone*
- *(Mostly) Improve the Microarchitecture*
  - *Pipelining, Caches*
  - *Branch Prediction, Speculative Execution*
  - *Out-of-order Execution, Trace Cache*
- *Today, we have too many transistors*
  - *Ergo, multicore (CMP)*

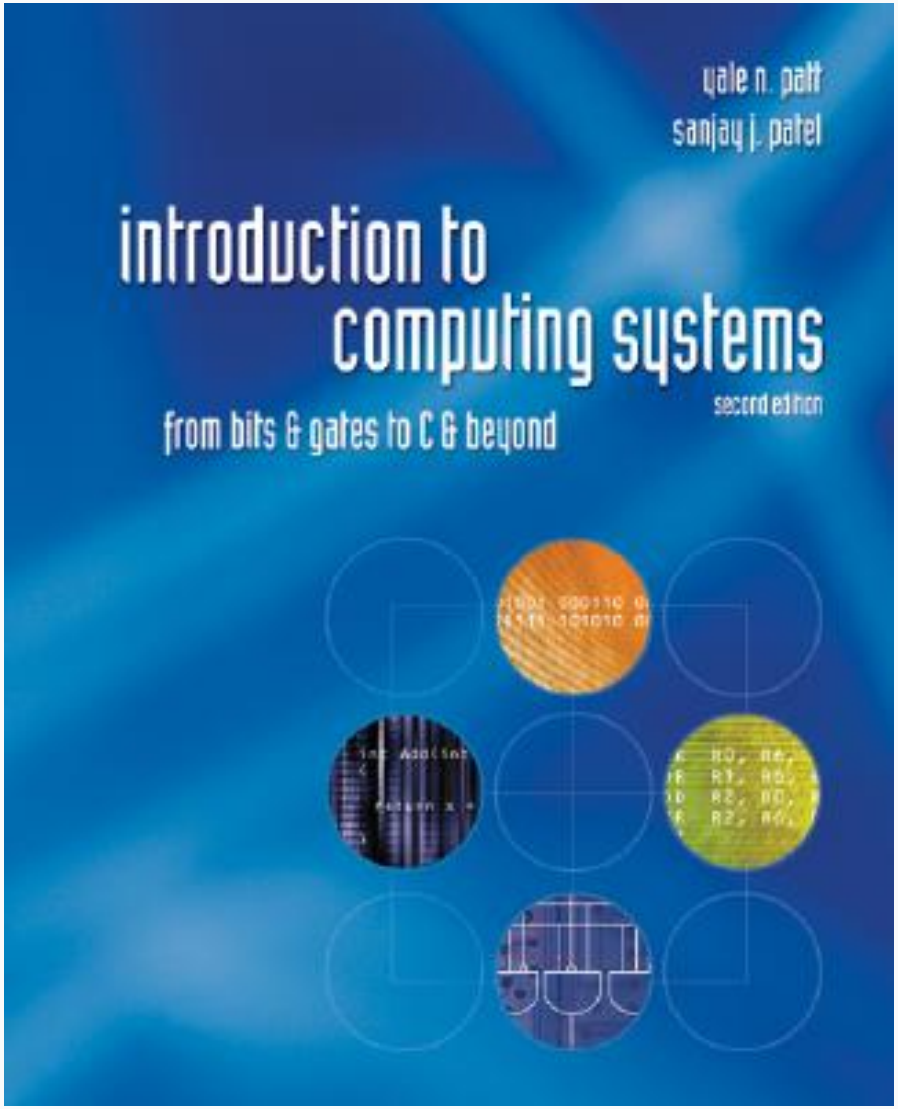
Intro: Takeaways



Yale N. Patt, *Microprocessor Performance, Phase 2: Can We Harness the Transformation Hierarchy*

http://hps.ece.utexas.edu/videos.html

Intro: Takeaways



Intro: The Times They Are a-Changin'

Intro: The Times They Are a-Changin'?

Intro: Modern Computer Architecture (cont'd)

Intro: Modern Computer Architecture (cont'd)

https://users.ece.cmu.edu/~omutlu/lecture-videos.html

Course Lecture Videos and Materials

- Undergraduate Computer Architecture Course Lecture Videos (2013, 2014, 2015)
- Undergraduate Computer Architecture Course Materials (2013, 2014, 2015)
- Graduate Computer Architecture Course Materials (Lecture Videos)
- Parallel Computer Architecture Course Materials (Lecture Videos)
- Memory Systems Short Course Materials (Lecture Video on Main Memory and DRAM Basics)

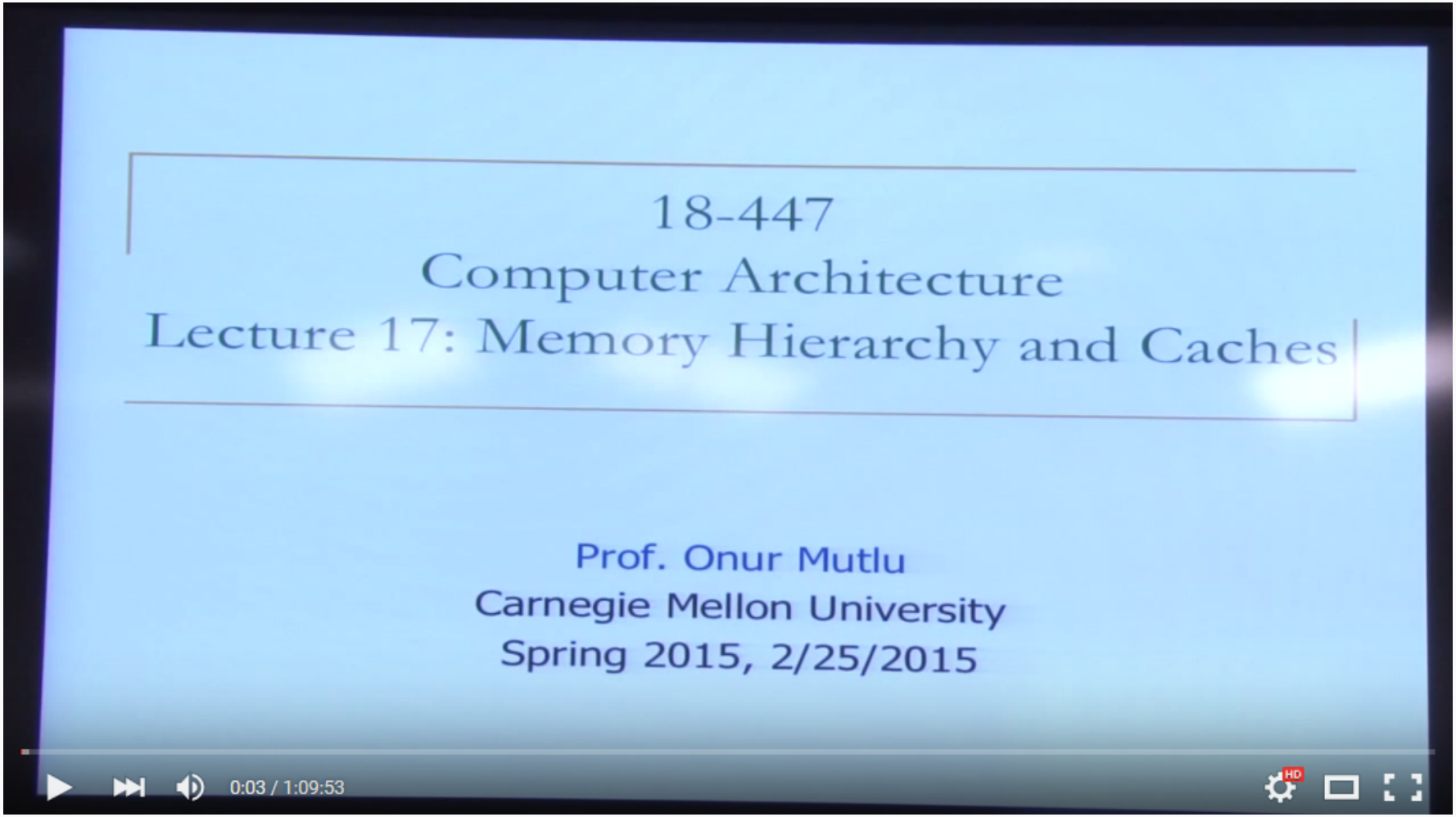
Talk Videos and Materials

- MemCon 2013 Lecture on "Memory Scaling" Slides (pptx) Slides (pdf) MemCon 2013 paper
- "Rethinking the Systems We Design" Lecture Video Slides (pptx) (pdf)

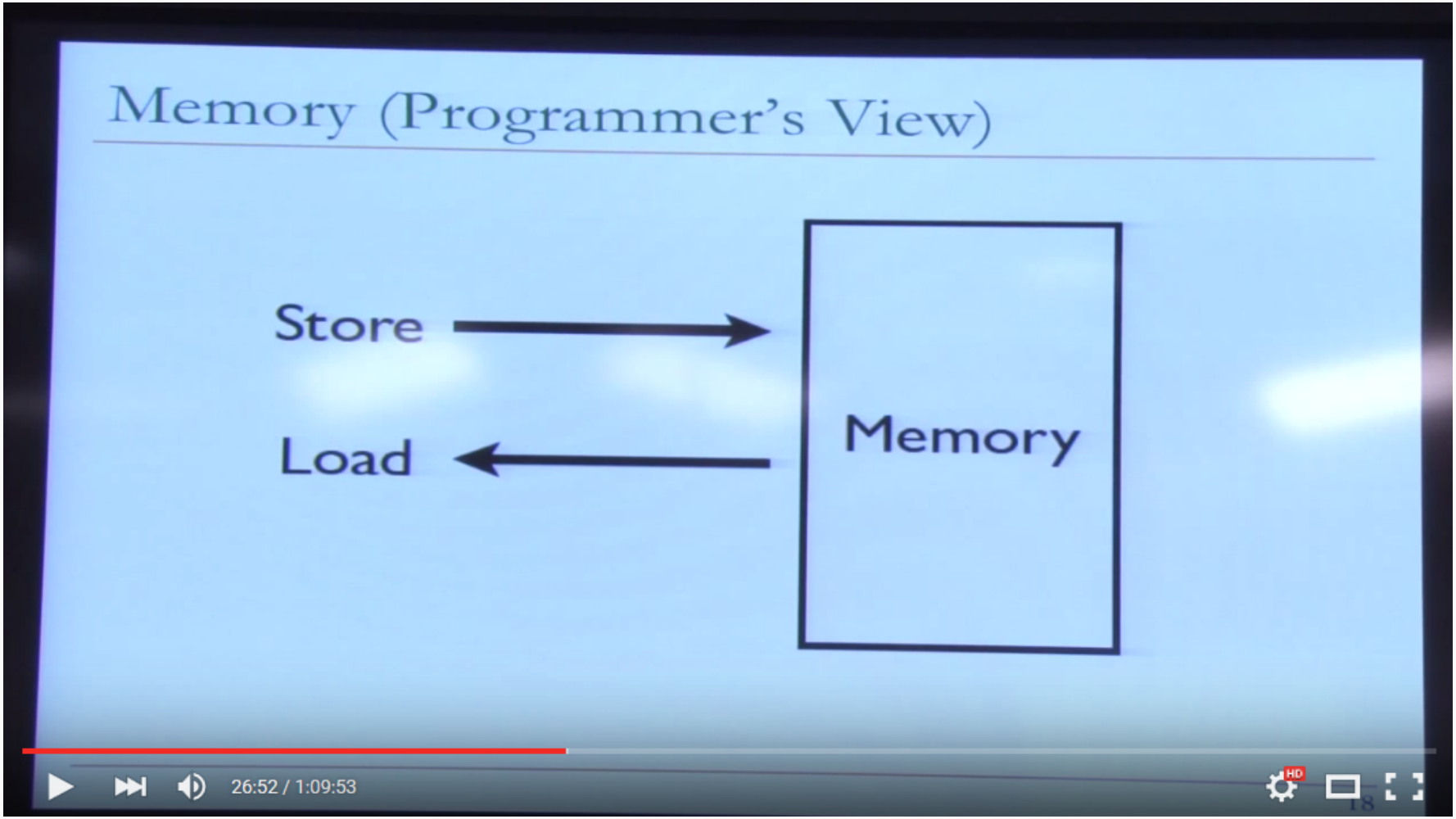
Other Open Source Materials

- Open Source Tools, Software and Data Sets

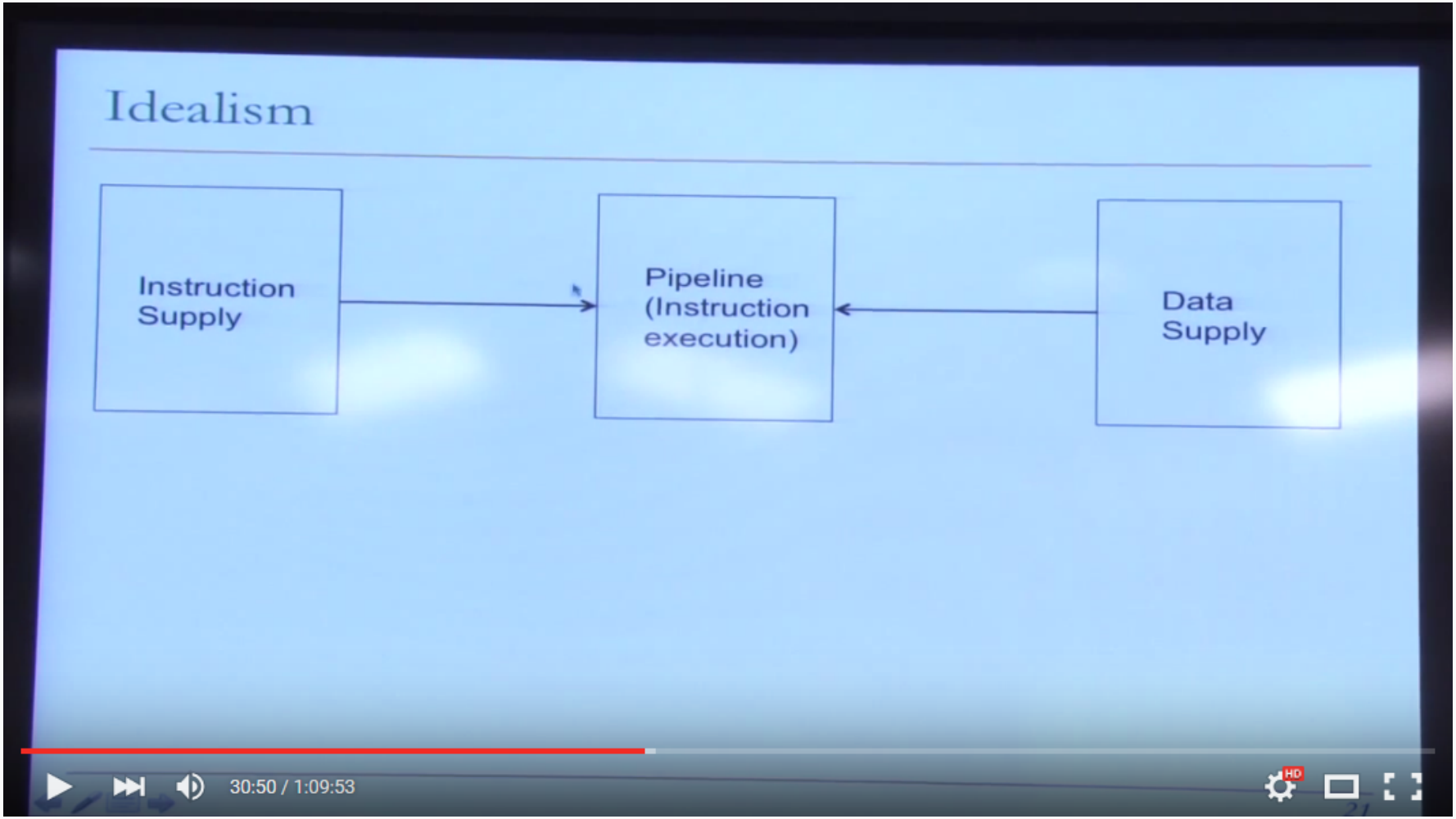
Memory Hierarchy and Caches - Prof. Onur Mutlu



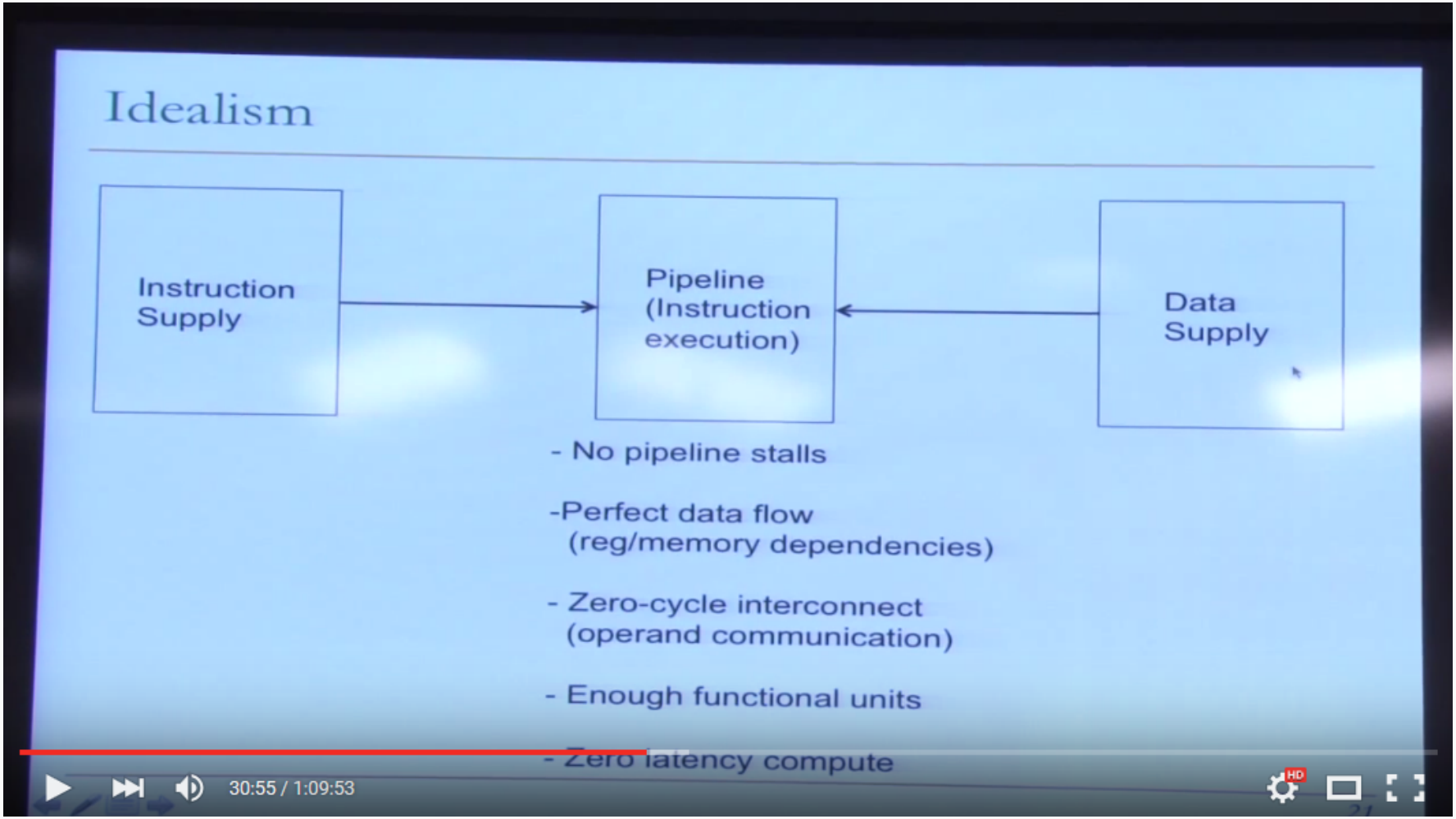
https://youtu.be/AXf_C5qOm8o



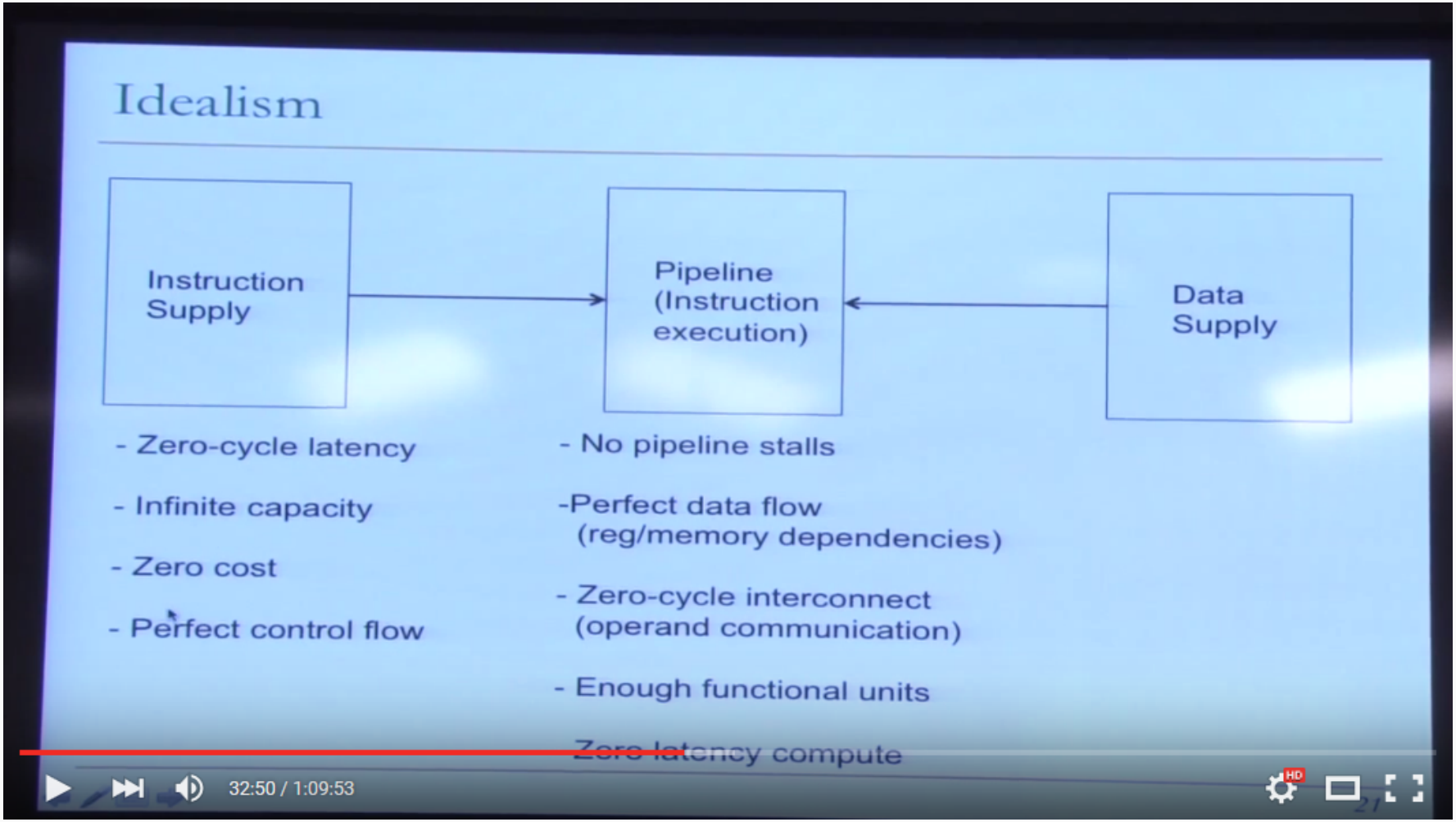
https://youtu.be/AXf_C5qOm8o



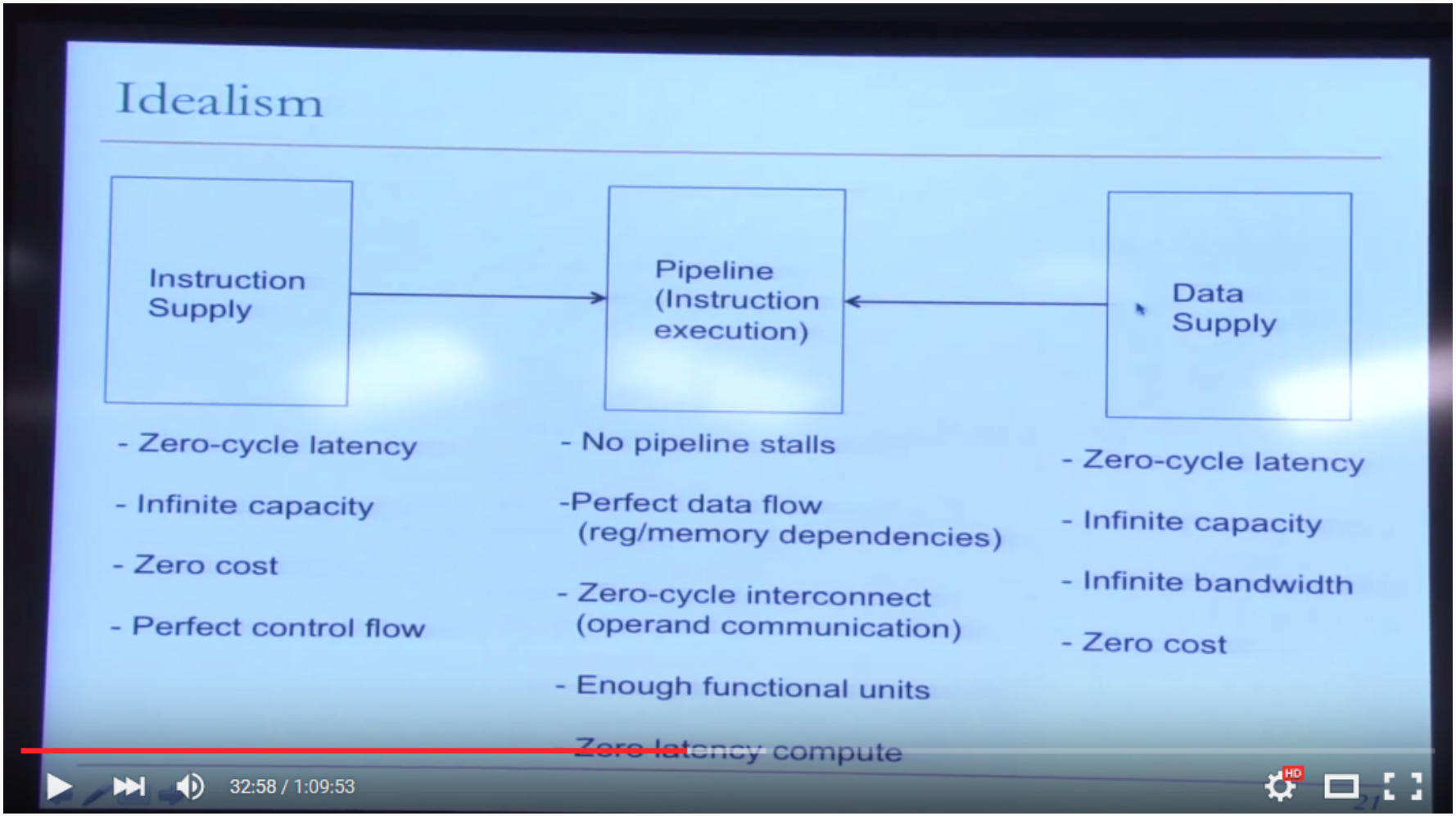
https://youtu.be/AXf_C5qOm8o



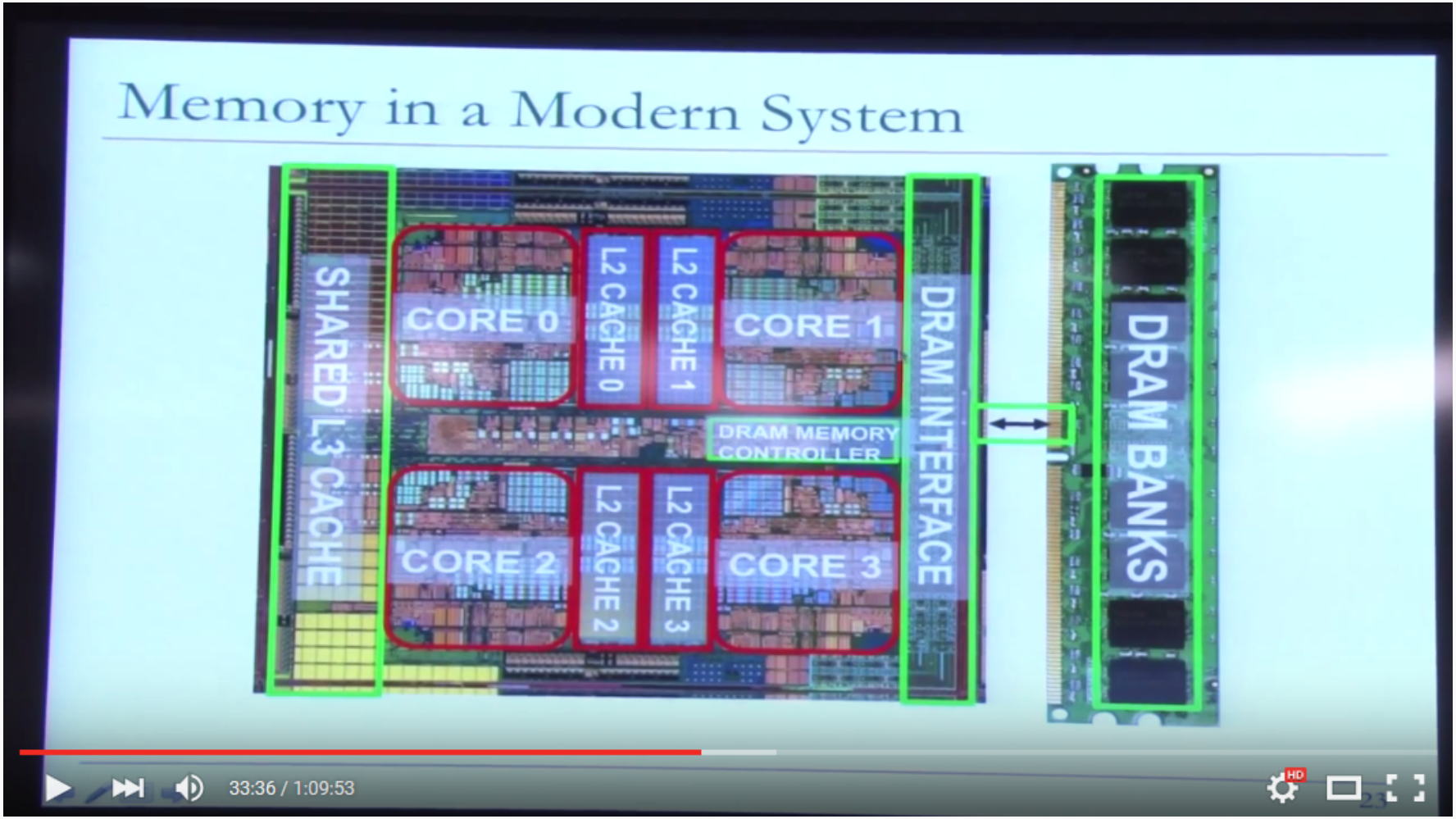
https://youtu.be/AXf_C5qOm8o



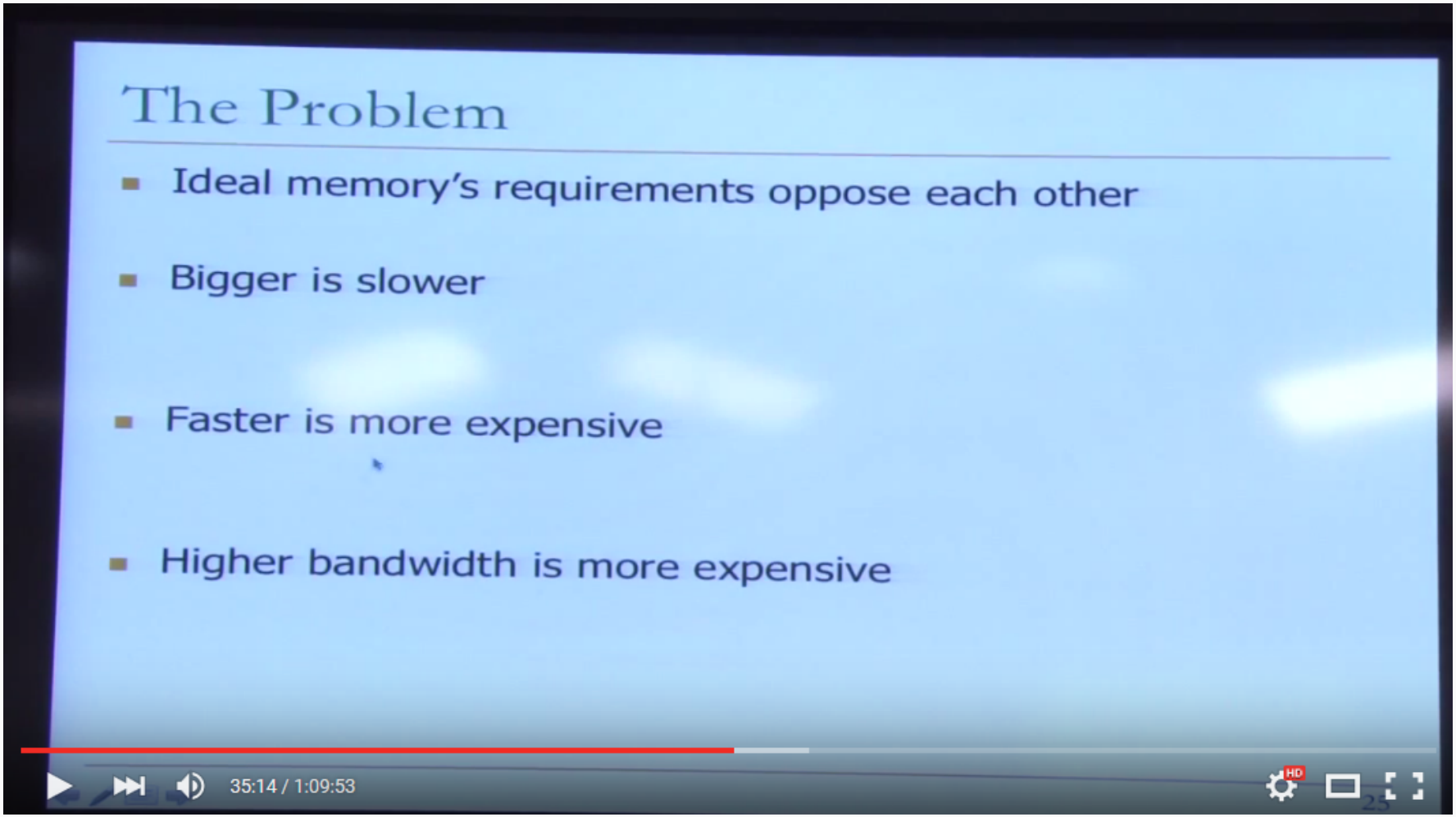
https://youtu.be/AXf_C5qOm8o



https://youtu.be/AXf_C5qOm8o



https://youtu.be/AXf_C5qOm8o



https://youtu.be/AXf_C5qOm8o

Intro - mIRC

mirc.com

Intro - IRC

Network Working Group J. Oikarinen
Request for Comments: 1459 D. Reed
May 1993

Internet Relay Chat Protocol

Status of This Memo

This memo defines an Experimental Protocol for the Internet community. Discussion and suggestions for improvement are requested. Please refer to the current edition of the "IAB Official Protocol Standards" for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

The IRC protocol was developed over the last 4 years since it was first implemented as a means for users on a BBS to chat amongst themselves. Now it supports a world-wide network of servers and clients, and is stringing to cope with growth. Over the past 2 years, the average number of users connected to the main IRC network has grown by a factor of 10.

The IRC protocol is a text-based protocol, with the simplest client being any socket program capable of connecting to the server.

Intro - Channels

An **IRC channel takeover** is an acquisition of IRC channel operator status by someone other than the channel's owner. It has largely been eliminated due to the increased use of services on IRC networks.

Contents [hide]

Riding the split [edit]

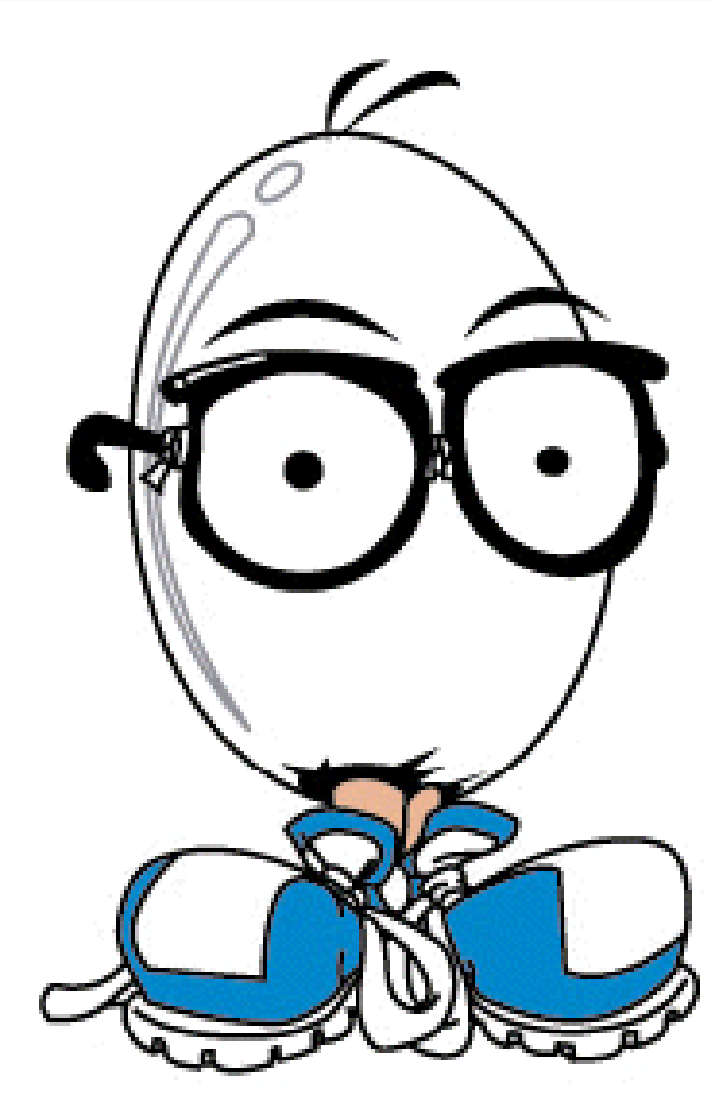
The most common variety of channel takeover uses disconnections caused by a netsplit; this is called **riding the split**. After such mass disconnections, a channel may be left without users, allowing the first rejoining user to recreate the channel and gain operator status. When the servers merge, any pre-existing operators retain their status, allowing the new user to kick out the original operators and take over the channel.

A simple prevention mechanism involves *timestamping* (abbreviated to *TS*), or checking the creation dates of the channels being merged. This was first implemented by Undernet (ircu) and is now common in many IRC servers. If both channels were created at the same time, all user statuses are retained when the two are combined; if one is newer than the other, special statuses are removed from those in the newer channel.

Additionally, a newer protection involving timestamping is used when a server splits away from the main network (when it no longer detects that IRC services are available), it disallows anyone creating a channel to be given operator privileges.

https://en.wikipedia.org/wiki/Internet_Relay_Chat_takeover

Intro - Eggdrop



http://www.eggheads.org/

src/main.c

```
int main(int arg_c, char **arg_v)
{
  // ...
  debug0("main: entering loop");
  while (1) {
    mainloop(1);
  }
}
```

src/main.c

```
int mainloop(int toplevel)
{
  // ...
  char buf[520];
  // ...
  xx = sockgets(buf, &i);
  // ...
  if (xx >= 0) {                /* Non-error */
    int idx;

    for (idx = 0; idx < dcc_total; idx++)
  // ...
}
```

Intro: (Recurring) Takeaways

- Hardware (still) matters for software
- Physics (still) matters for hardware

Intro: (Recurring) Takeaways

Yale N. Patt at Yale Patt 75 Visions of the Future Computer Architecture Workshop:

” ’Are you a software person or a hardware person?’
I’m a person
this pigeonholing has to go

We must break the layers

Abstractions are great
- AFTER you understand what’s being abstracted”

Yale N. Patt, 2013 IEEE CS Harry H. Goode Award Recipient Interview — https://youtu.be/S7wXivUy-tk
Yale N. Patt at Yale Patt 75 Visions of the Future Computer Architecture Workshop — https://youtu.be/x4LH1cJCvxs

Intro: (Recurring) Takeaways

"Finally, it is also very fortunate to see from a researcher's point of view that many open and fundamental questions will definitely appear and that these will stimulate and keep our lives busy, hopefully for the next 100 years."

Hardware/Software Codesign: The Past, the Present, and Predicting the Future
http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6172642

Task(s)

- Get some numbers,
- crunch them,
- get some more

Task(s)

```
do
{
```

- Get some numbers,
- crunch them,
- get some more

```
}
while (more_numbers);
```

Task(s)

```
do
{
```

- Get some numbers,
  - network
- crunch them,
  - incremental, online
- get some more

```
}
while (more_numbers);
```

Numbers

Prices						
Date	Open	High	Low	Close	Volume	Adj Close*
Sep 24, 2015	28.38	28.67	27.87	28.48	32,437,500	28.48
Sep 23, 2015	29.02	29.09	28.59	28.74	25,770,000	28.74
Sep 22, 2015	28.67	28.90	28.48	28.67	26,593,200	28.67
Sep 21, 2015	29.09	29.34	28.94	29.16	24,179,500	29.16
Sep 18, 2015	29.22	29.42	28.81	29.02	60,138,700	29.02
Sep 17, 2015	29.62	30.20	29.55	29.71	29,468,900	29.71
Sep 16, 2015	29.62	29.79	29.40	29.77	25,932,000	29.77
Sep 15, 2015	29.56	29.94	29.48	29.73	29,399,900	29.73
Sep 14, 2015	29.47	29.53	29.09	29.39	24,063,700	29.39

Numbers

Prices						
Date	Open	High	Low	Close	Volume	Adj Close*
Mar 5, 2001	29.88	30.75	29.50	30.38	33,275,700	22.03
Mar 2, 2001	28.50	31.13	28.38	29.31	54,626,200	21.26
Mar 1, 2001	28.25	29.25	27.06	29.13	56,268,300	21.12
Feb 28, 2001	29.38	30.19	28.00	28.56	48,601,400	20.71
Feb 27, 2001	29.50	30.88	29.00	29.00	40,787,300	21.03
Feb 26, 2001	30.56	30.70	28.44	29.50	60,349,600	21.39
Feb 23, 2001	30.38	30.75	28.50	29.94	63,558,300	21.71
Feb 22, 2001	30.94	31.56	29.63	30.00	58,517,400	21.76
Feb 21, 2001	31.00	32.75	30.50	30.75	51,582,200	22.30

Numbers

```
"05Mar2001","3:09:00 PM",30.28125
"05Mar2001","3:10:00 PM",30.34375
"05Mar2001","3:11:00 PM",30.21875
"05Mar2001","3:12:00 PM",30.28125
"05Mar2001","3:13:00 PM",30.28125
"05Mar2001","3:14:00 PM",30.28125
"05Mar2001","3:15:00 PM",30.28125
"05Mar2001","3:16:00 PM",30.28125
"05Mar2001","3:20:00 PM",30.34375
"05Mar2001","3:21:00 PM",30.34375
"05Mar2001","3:22:00 PM",30.34375
```

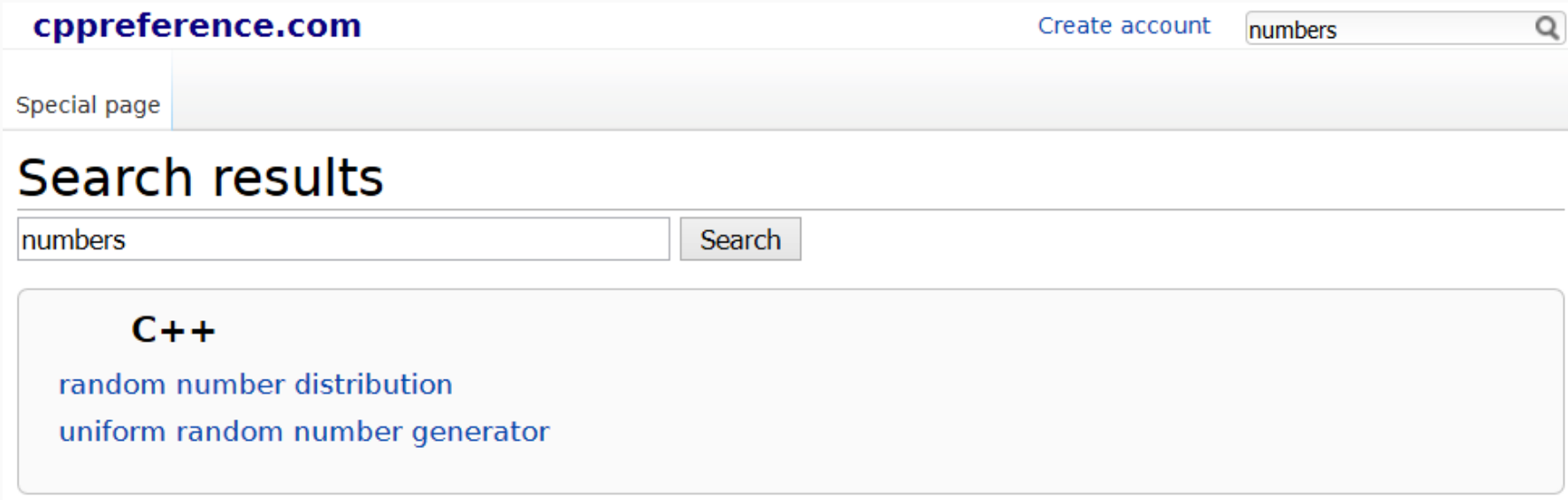
Numbers

TIME	PRICE	SHARES
16:11:27	28.45	1,000
15:59:59	28.48	100
15:59:59	28.48	23
15:59:59	28.48	200
15:59:59	28.48	100
15:59:59	28.48	100
15:59:58	28.48	100
15:59:58	28.48	49
15:59:58	28.48	400
15:59:58	28.48	100

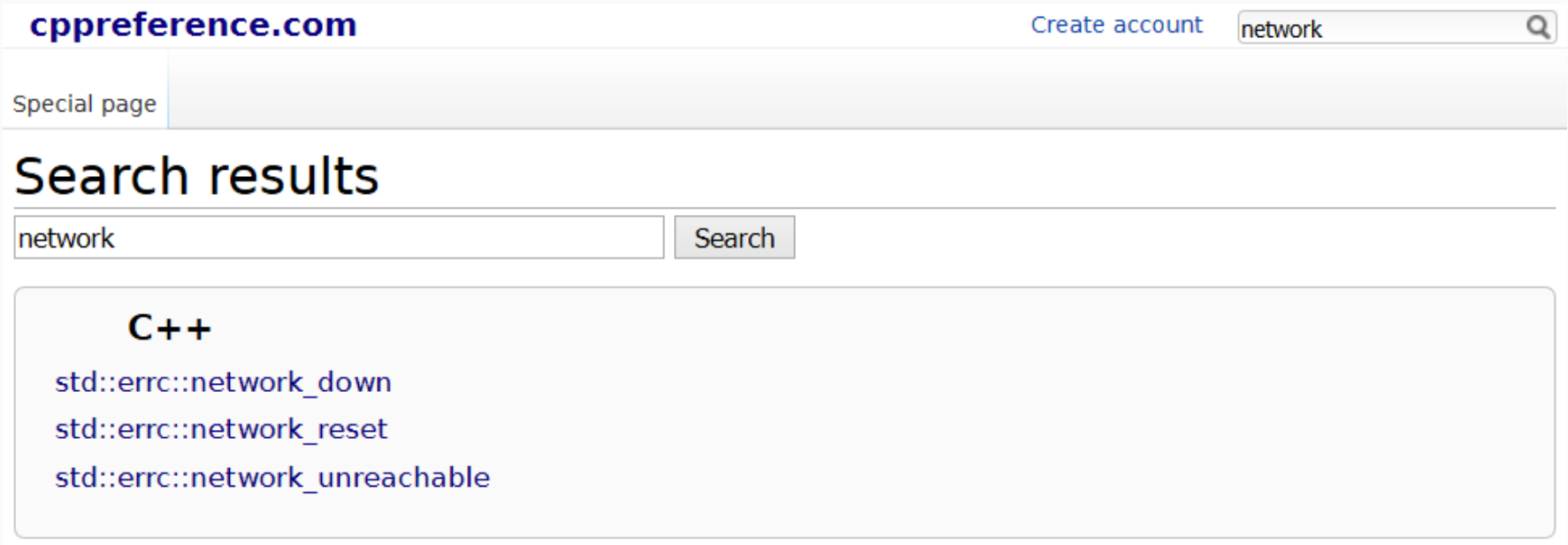
Numbers -> C++ ?

isocpp.org

Numbers -> C++ ?


cppreference.com
Create account
numbers
Special page
Search results
numbers
Search
C++
random number distribution
uniform random number generator

Numbers -> C++ ?


cppreference.com
Create account
network
Special page
Search results
network
Search
C++
std::errc::network_down
std::errc::network_reset
std::errc::network_unreachable

Numbers -> C++ ?

ISO/IEC DTS 19216:xxxx	Networking TS	In development, Draft n4478 (2015-04-13)

http://en.cppreference.com/w/cpp/experimental

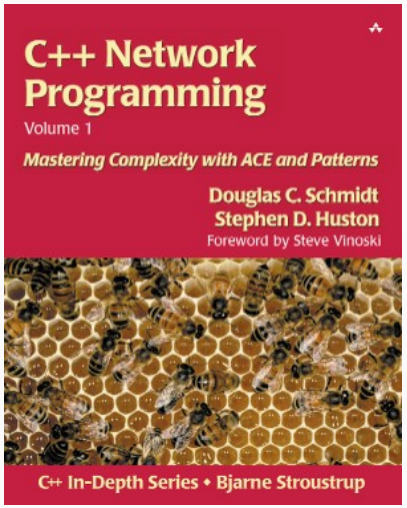
Numbers -> C++ ?

Communication

- C++ RESTful framework - C++ micro-framework designed to be embedded into a wide range of applications.
- C++ REST SDK - asynchronous HTTP client and listener, asynchronous Stream, URI, JSON
- cpr - a modern C++ HTTP requests library
- cpp-netlib - cpp-netlib: The C++ Network Library
- Boost.Asio - asynchronous and synchronous networking, timers, serial I/O
- POCO - networking: encryption, HTTP; Zip files
- ACE - asynchronous networking, event demultiplexing, messaging, CORBA
- wvstreams
- gsoap
- Unicomm - asynchronous networking, high-level TCP communication framework
- restful_mapper - ORM for consuming RESTful JSON APIs in C++
- zeromq - fast message queue
- curlpp - C++ wrapper for CURL library
- Apache Thrift - The Apache Thrift software framework, for scalable cross-language services development, combines a software stack with a code generation engine to build services that work efficiently and seamlessly between C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, OCaml and Delphi and other languages.
- libashttp - asynchronous HTTP client library
- Simple C++ REST library - Very simple and self documenting library for creating a REST API in your c++ application
- libtins - Network packet crafting and sniffing library
- HTTPP - Simple, production ready HTTP server built on top of Boost and a client built on top of libcurl. (BSD)

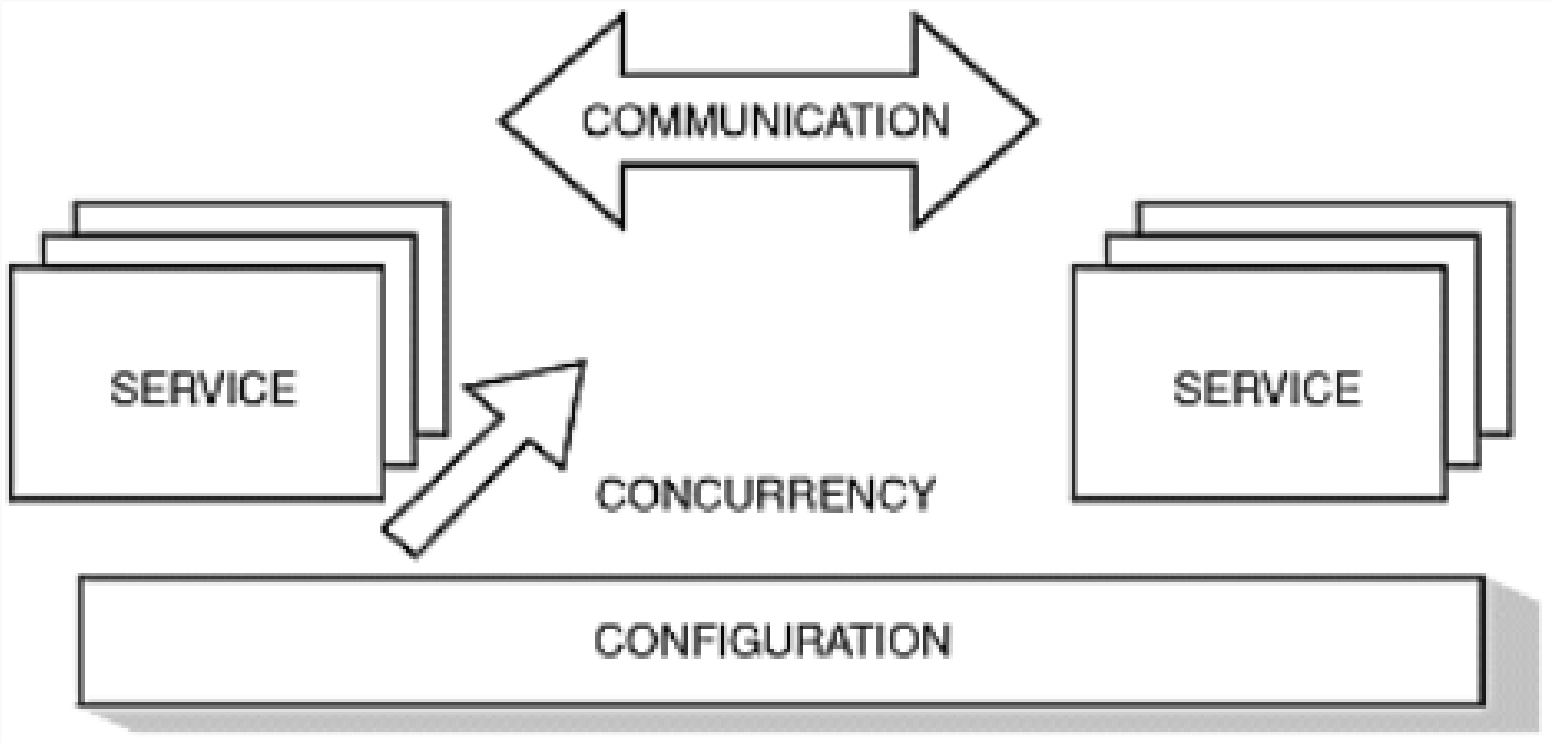
http://en.cppreference.com/w/cpp/links/libs#Communication

ADAPTIVE Communication Environment (ACE)



http://www.cs.wustl.edu/~schmidt/ACE/book1/

Networked Application Design Dimensions



C++NPv1

Networked Application Design Dimensions

dimensions is composed of a set of relatively independent alternatives. Although mostly orthogonal to each other, changes to one or more dimensions of your networked application can change its "shape" accordingly. Design changes therefore don't occur in isolation. Keep this in mind as you consider the following design dimensions:

1. **Communication dimensions** address the rules, form, and level of abstraction that networked applications use to interact.
2. **Concurrency dimensions** address the policies and mechanisms governing the proper use of processes and threads to represent multiple service instances, as well as how each service instance may use multiple threads internally.
3. **Service dimensions** address key properties of a networked application service, such as the duration and structure of each service instance.
4. **Configuration dimensions** address how networked services are iden-tified and the time at which they are bound together to form complete applications. Configuration dimensions often affect more than one service, as well as the relationships between services.

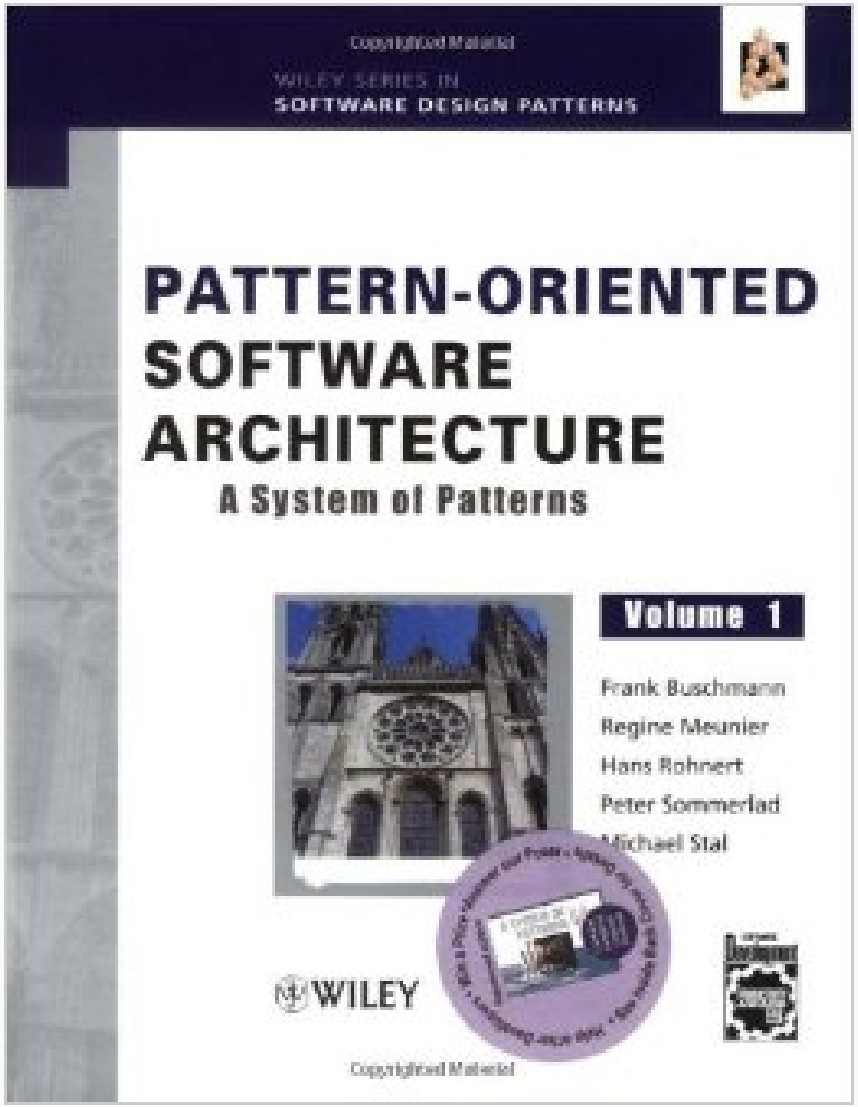
http://www.informit.com/articles/article.aspx?p=25486&seqNum=3

Pattern-Oriented Software Architecture (POSA)

http://www.cs.wustl.edu/~schmidt/POSA/

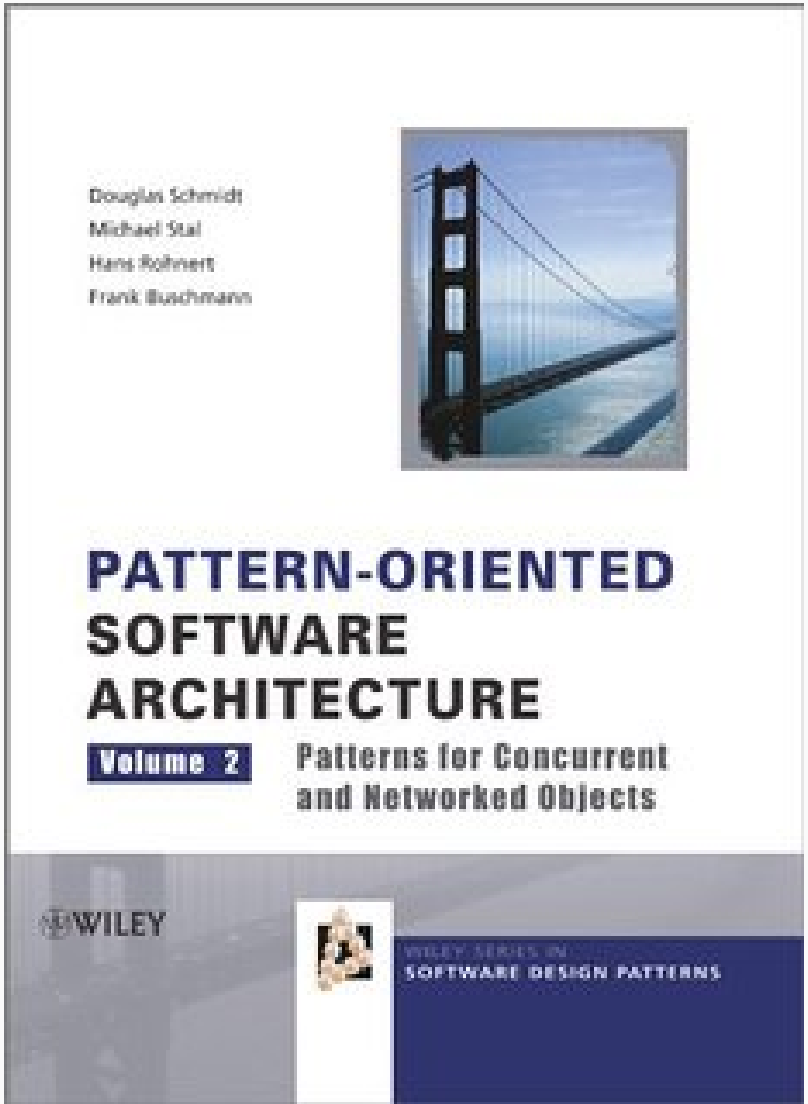
- Pattern-Oriented Software Architecture: A System of Patterns, Volume 1
- Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects, Volume 2

Pipes and Filters



Challenges of Concurrent and Networked Software

Concurrency Patterns



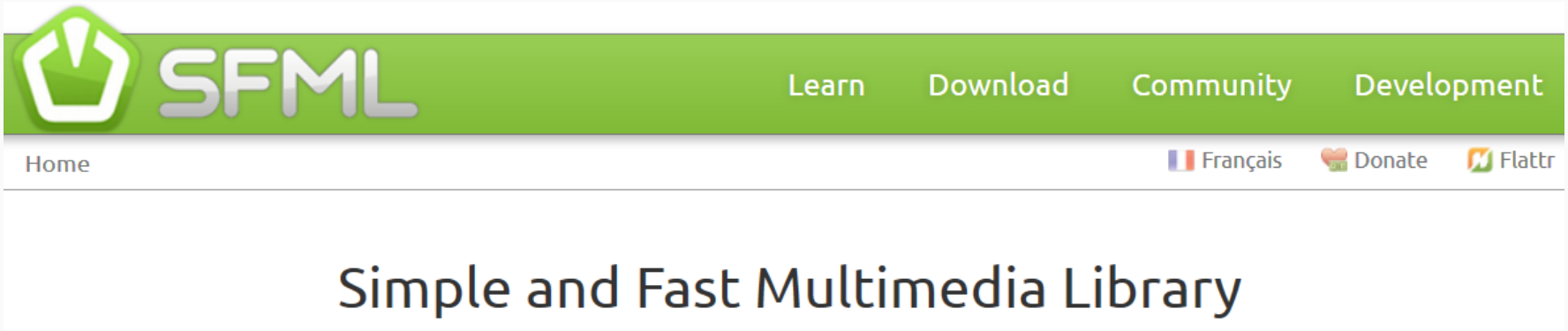
Excuses

- Make It Work
- Make It Right
- Make It Fast
  http://c2.com/cgi/wiki?MakeItWorkMakeItRightMakeItFast
- Do The Simplest Thing That Could Possibly Work
  http://c2.com/cgi/wiki?DoTheSimplestThingThatCouldPossiblyWork

~~Excuses~~ Design Principles

- Make It Work
- Make It Right
- Make It Fast
  http://c2.com/cgi/wiki?MakeItWorkMakeItRightMakeItFast
- Do The Simplest Thing That Could Possibly Work
  http://c2.com/cgi/wiki?DoTheSimplestThingThatCouldPossiblyWork

SFML



http://sfml-dev.org/

SFML is multi-platform

With SFML, your application can compile and run out of the box on the most common operating systems: Windows, Linux, Mac OS X and soon Android & iOS.

Pre-compiled SDKs for your favorite OS are available on the download page.

http://sfml-dev.org/

Download SFML 2.3.2

Windows	Visual C++ 10 (2010) - 32-bit	Download | 11.9 MB	Visual C++ 10 (2010) - 64-bit	Download | 13.2 MB
	Visual C++ 11 (2012) - 32-bit	Download | 13.4 MB	Visual C++ 11 (2012) - 64-bit	Download | 15.0 MB
	Visual C++ 12 (2013) - 32-bit	Download | 12.8 MB	Visual C++ 12 (2013) - 64-bit	Download | 14.3 MB
	GCC 4.7.1 TDM (SJLJ) - 32-bit	Download | 13.5 MB	GCC 4.7.1 TDM (SJLJ) - 64-bit	Download | 16.3 MB
	GCC 4.8.1 TDM (SJLJ) - 32-bit	Download | 13.3 MB	GCC 4.8.1 TDM (SJLJ) - 64-bit	Download | 15.3 MB
	GCC 4.9.2 MinGW (DW2) - 32-bit	Download | 13.6 MB	GCC 4.9.2 MinGW (SEH) - 64-bit	Download | 14.5 MB

On Windows, choosing 32 or 64-bit libraries should be based on which platform you want to compile for, not which OS you have. Indeed, you can perfectly compile and run a 32-bit program on a 64-bit Windows. So you'll most likely want to target 32-bit platforms, to have the largest possible audience. Choose 64-bit packages only if you have good reasons.

Linux	GCC - 32-bit	Download | 1.9 MB	GCC - 64-bit	Download | 1.9 MB

On Linux, if you have a 64-bit OS then you have the 64-bit toolchain installed by default. Compiling for 32-bit is possible, but you have to install specific packages and/or use specific compiler options to do so. So downloading the 64-bit libraries is the easiest solution if you're on a 64-bit Linux.

Mac OS X	Clang - universal 32+64-bit (OS X 10.7+, compatible with C++11 and libc++)	Download | 6.05 MB
	Mac OS X libraries are universal: they contain both the 32 and 64-bit versions of SFML and can therefore be used for both 32 and 64-bit builds.	

All	Source code	Download | 21.5 MB
	HTML Documentation	Download | 1.38 MB

```
// prepare the request
// STD 66: Syntax Components
// https://en.wikipedia.org/wiki/URI scheme#Generic syntax
namespace api = yahoo;
sf::Http http{api::host()};
const auto uri = api::path() + api::symbol_query(symbol, start_date, end_date);
```

https://github.com/philsquared/Catch

Catch: C++ Automated Test Cases in Headers I

```
#include "catch.hpp"
```

(also shown on this slide: full list of dependencies and complete build instructions)

Catch: C++ Automated Test Cases in Headers II

```
SCENARIO("API symbol query constructed correctly", "[symbol][query][unit]")
{
  GIVEN("Query date parameters")
  {
    const date start_date = ...#1;
    const date end_date = ...#2;

    WHEN("the symbol is set to X")
    {
      const symbol_type symbol = "X";
      const auto uri = api::path() + api::symbol_query(symbol, start_date,

      THEN("the built URI is correct")
      {
        REQUIRE(uri == ...X...#1...#2);
      }
    }
  }
}
```

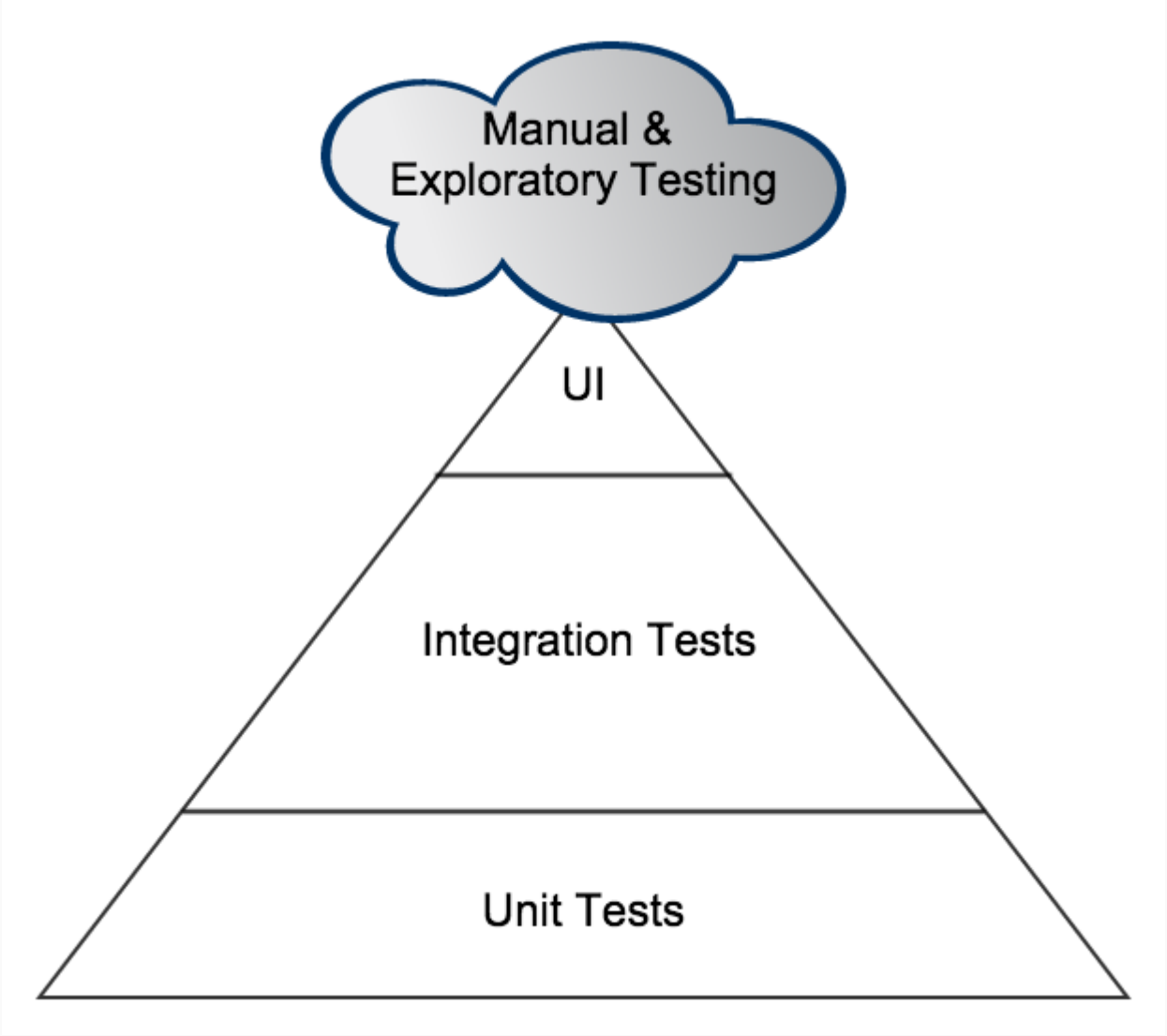
Testing?

These things are "easy mode" for tests. -- Ben Deane

https://github.com/boostcon/cppnow_presentations_2015/raw/master/files/testing-battlenet.pdf

https://cppcon2015.sched.org/event/ac2534ecb08510c5810e7df34cdddb94

Testing



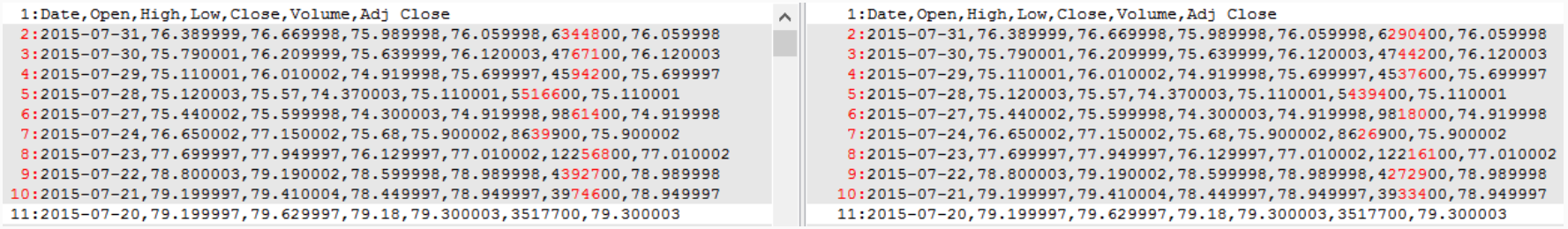
http://qa-matters.com/2014/12/28/layers-of-test-automation/

Testing: Differential, Integration, End-to-End

Name	Size	Date	<=>	Date	Size	Name
AAPL.csv	574,186	09/07/15 12:53:21	=	09/07/15 11:24:57	574,186	AAPL.csv
AXP.csv	742,258	09/07/15 13:22:05	≠	09/07/15 13:05:38	742,258	AXP.csv
BA.csv	880,857	08/30/15 10:43:42	=	08/30/15 10:43:39	880,857	BA.csv
CAT.csv	843,818	09/01/15 12:42:20	=	09/07/15 13:06:39	843,818	CAT.csv
CSCO.csv	406,832	09/07/15 12:54:34	=	09/07/15 11:24:57	406,832	CSCO.csv
CVX.csv	698,702	08/30/15 10:43:45	=	08/30/15 10:43:39	698,702	CVX.csv
DD.csv	924,423	09/07/15 13:22:12	=	08/30/15 10:43:39	924,423	DD.csv
DIS.csv	923,921	09/07/15 13:22:14	≠	09/07/15 13:22:01	923,843	DIS.csv
GE.csv	885,167	09/07/15 13:13:49	=	09/07/15 12:53:20	885,167	GE.csv
GS.csv	284,695	08/30/15 10:43:48	=	08/30/15 10:43:38	284,695	GS.csv
HD.csv	565,154	09/07/15 12:53:38	=	09/07/15 11:24:57	565,154	HD.csv
IBM.csv	873,562	08/30/15 10:43:49	=	08/30/15 10:43:39	873,562	IBM.csv
INTC.csv	568,053	08/30/15 10:43:50	=	08/30/15 10:43:38	568,053	INTC.csv
JNJ.csv	736,228	08/30/15 10:43:51	=	09/07/15 12:53:20	736,228	JNJ.csv
JPM.csv	508,440	09/01/15 12:42:59	=	09/07/15 12:57:13	508,440	JPM.csv
KO.csv	878,394	09/07/15 13:22:30	≠	09/07/15 12:54:22	878,394	KO.csv
MCD.csv	728,525	08/30/15 10:43:53	=	08/30/15 10:43:39	728,525	MCD.csv
MMM.csv	710,500	08/30/15 10:43:54	=	08/30/15 10:43:39	710,500	MMM.csv
MRK.csv	745,452	09/07/15 12:53:53	=	09/07/15 13:05:39	745,452	MRK.csv
MSFT.csv	482,863	08/30/15 10:43:55	=	08/30/15 10:43:38	482,863	MSFT.csv
NKE.csv	565,090	09/07/15 12:53:56	=	09/07/15 11:24:57	565,090	NKE.csv
PFE.csv	703,449	08/30/15 10:43:57	=	08/30/15 10:43:39	703,449	PFE.csv
PG.csv	732,842	09/07/15 13:06:16	=	09/07/15 11:24:58	732,842	PG.csv
TRV.csv	440,107	09/01/15 12:48:51	=	09/07/15 13:22:00	440,107	TRV.csv
UNH.csv	411,696	09/07/15 12:54:03	=	08/30/15 12:10:58	411,696	UNH.csv
UTX.csv	729,137	08/30/15 10:43:59	=	09/07/15 12:53:19	729,137	UTX.csv
V.csv	129,884	08/30/15 10:44:00	=	08/30/15 10:43:38	129,884	V.csv
VZ.csv	547,934	09/07/15 13:22:48	=	09/07/15 12:53:19	547,934	VZ.csv
WMT.csv	681,411	08/30/15 10:44:01	=	08/30/15 10:43:39	681,411	WMT.csv
XOM.csv	719,894	08/30/15 10:44:02	=	08/30/15 10:43:39	719,894	XOM.csv

http://www.hpl.hp.com/hpjournal/dtj/vol10num1/vol10num1art9.pdf

Testing: Differential, Integration, End-to-End



```
1:Date,Open,High,Low,Close,Volume,Adj Close
2:2015-07-31,76.389999,76.669998,75.989998,76.059998,6344800,76.059998
3:2015-07-30,75.790001,76.209999,75.639999,76.120003,4767100,76.120003
4:2015-07-29,75.110001,76.010002,74.919998,75.699997,4594200,75.699997
5:2015-07-28,75.120003,75.57,74.370003,75.110001,5516600,75.110001
6:2015-07-27,75.440002,75.599998,74.300003,74.919998,9861400,74.919998
7:2015-07-24,76.650002,77.150002,75.68,75.900002,8639900,75.900002
8:2015-07-23,77.699997,77.949997,76.129997,77.010002,12256800,77.010002
9:2015-07-22,78.800003,79.190002,78.599998,78.989998,4392700,78.989998
10:2015-07-21,79.199997,79.410004,78.449997,78.949997,3974600,78.949997
11:2015-07-20,79.199997,79.629997,79.18,79.300003,3517700,79.300003
```

```
1:Date,Open,High,Low,Close,Volume,Adj Close
2:2015-07-31,76.389999,76.669998,75.989998,76.059998,6290400,76.059998
3:2015-07-30,75.790001,76.209999,75.639999,76.120003,4744200,76.120003
4:2015-07-29,75.110001,76.010002,74.919998,75.699997,4537600,75.699997
5:2015-07-28,75.120003,75.57,74.370003,75.110001,5439400,75.110001
6:2015-07-27,75.440002,75.599998,74.300003,74.919998,9818000,74.919998
7:2015-07-24,76.650002,77.150002,75.68,75.900002,8626900,75.900002
8:2015-07-23,77.699997,77.949997,76.129997,77.010002,12216100,77.010002
9:2015-07-22,78.800003,79.190002,78.599998,78.989998,4272900,78.989998
10:2015-07-21,79.199997,79.410004,78.449997,78.949997,3933400,78.949997
11:2015-07-20,79.199997,79.629997,79.18,79.300003,3517700,79.300003
```

https://www.symphonious.net/2015/04/30/making-end-to-end-tests-work/

?

?

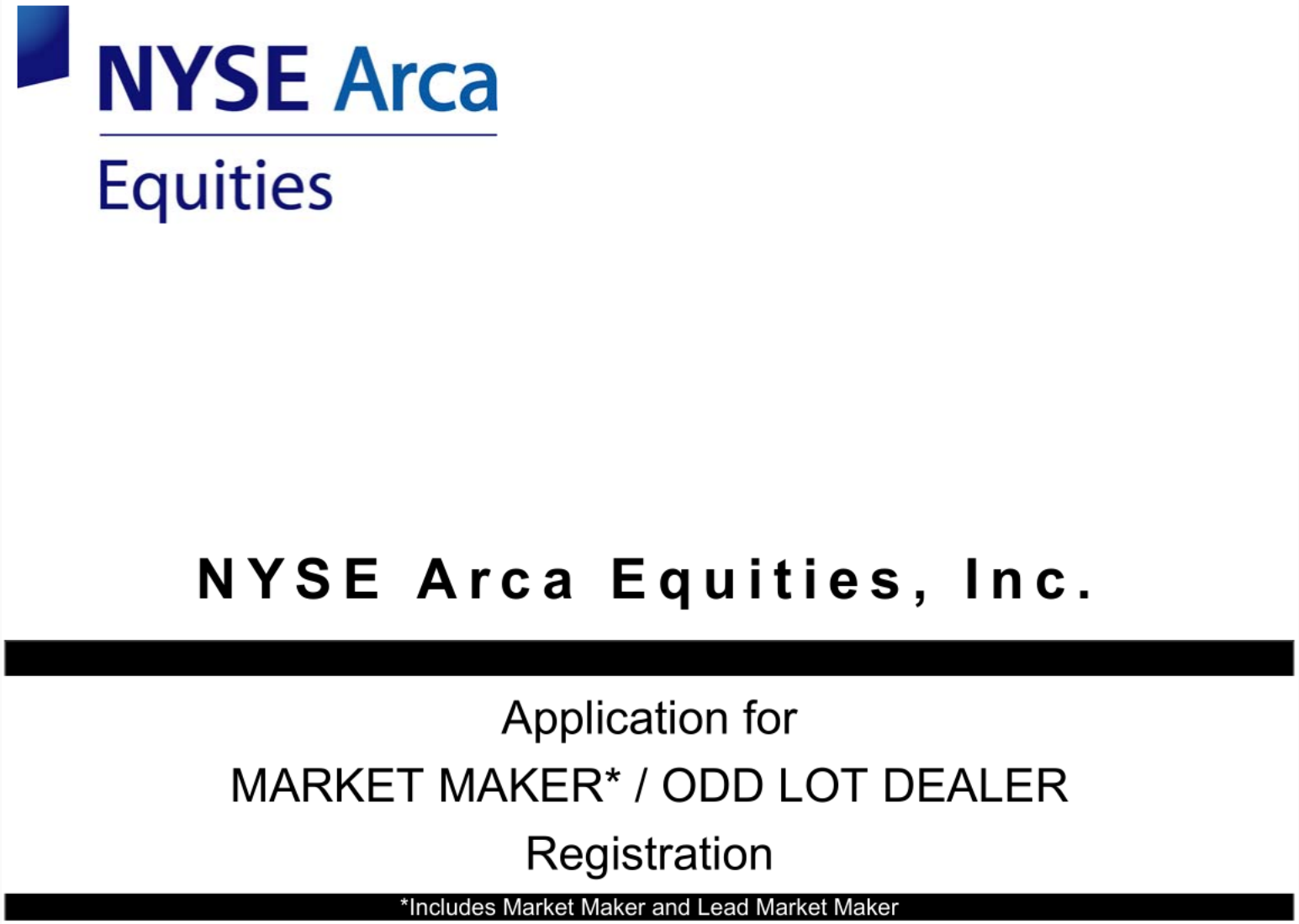
Fundamental theorem of software engineering

The **fundamental theorem of software engineering** (FTSE) is a term originated by Andrew Koenig to describe a remark by Butler Lampson[1] attributed to the late David J. Wheeler:[2]

"We can solve any problem by introducing an extra level of indirection."

https://en.wikipedia.org/wiki/Fundamental_theorem_of_software_eng

NYSE ARCA TEST STOCK (ZVV)



http://www.archipelago.com/docs/NYSEArcaMM-LMMApp.pdf

NYSE ARCA test stock (ZVV)

Symbols – refers to stock symbols that may be assigned to a MMAT. Symbols are assigned at the firm level and are traded by the MMATs. Symbols may be added or removed on a daily basis, based on requests emailed to PCX Market Management, MM@PacificEx.com, with said request. At this time, OTC symbols are not eligible for market making. Upon approval, by default, two test symbols (ZVV and M.TEST) will be assigned to the firm for testing purposed.

http://www.archipelago.com/docs/NYSEArcaMM-LMMApp.pdf

NYSE ARCA test stock (ZVV)

```cpp
TEST_CASE("NYSE ARCA test stock (ZVV)", "[net][yahoo][integration]")
{
  const auto symbol = "ZVV";
  std::ostringstream output;
  namespace api = net::yahoo;

  const auto download_size = net::download_http(symbol, output, api::date{ "2012", "6", "19
  std::cout << "download_size = " << download_size << '\n';
  REQUIRE(download_size == 162);
  // 162 bytes:
  // Date,Open,High,Low,Close,Volume,Adj Close
  // 2012-09-21,2.77,2.77,2.77,2.77,000,2.77
  // 2012-09-20,2.77,2.77,2.77,2.77,000,2.77
  // 2012-09-19,2.77,2.77,2.77,2.77,000,2.77

  std::istringstream input(output.str());
  std::string line;
  std::getline(input, line); REQUIRE(line == "Date,Open,High,Low,Close,Volume,Adj Close");
  std::getline(input, line); REQUIRE(line == "2012-09-21,2.77,2.77,2.77,2.77,000,2.77");
  std::getline(input, line); REQUIRE(line == "2012-09-20,2.77,2.77,2.77,2.77,000,2.77");
  std::getline(input, line); REQUIRE(line == "2012-09-19,2.77,2.77,2.77,2.77,000,2.77");
}
```

NET::DOWNLOAD_HTTP -> NET::DOWNLOAD_SOCKET

```
char received_data[1024];
std::size_t received_bytes = 0;
std::size_t received_bytes_total = 0;

while (socket.receive(received_data, sizeof(received_data), received_bytes) == sf::Socket::Done)
{
  // started receiving
  if (current_state == message_state::pre_receive) current_state = message_state::status_line;
  //std::clog << "Receiving " << received_bytes << " bytes" << std::endl;
  boost::string_ref currently_received(received_data, received_bytes);
  current_state = discard_until_body(current_state, currently_received);
```

- cf. http://www.sfml-dev.org/tutorials/2.3/network-socket.php#non-blocking-sockets
- http://boost.org/libs/utility
  http://www.boost.org/doc/libs/master/libs/utility/doc/html/string_ref.html
  http://theboostcpplibraries.com/boost.utility

```
// Constructs from a NULL-terminated string
basic_string_ref(const charT* str);

// Constructs from a pointer, length pair
basic_string_ref(const charT* str, size_type len);
```

Boost.StringRef -- std::string_view

Marshall Clow: string_view - when to use it, and when not.

http://www.boost.org/doc/libs/release/libs/utility/doc/html/string_r

http://en.cppreference.com/w/cpp/experimental/basic_string_view

Boost.StringRef is an implementation of Jeffrey Yaskin's N3442: string_ref: a non-owning reference to a string.

When you are parsing/processing strings from some external source, frequently you want to pass a piece of text to a procedure for specialized processing. The canonical way to do this is as a `std::string`, but that has certain drawbacks:

1) If you are processing a buffer of text (say a HTTP response or the contents of a file), then you have to create the string from the text you want to pass, which involves memory allocation and copying of data.

2) if a routine receives a constant `std::string` and wants to pass a portion of that string to another routine, then it must create a new string of that substring.

3) A routine receives a constant `std::string` and wants to return a portion of the string, then it must create a new string to return.

`string_ref` is designed to solve these efficiency problems. A `string_ref` is a read-only reference to a contiguous sequence of characters, and provides much of the functionality of `std::string`. A `string_ref` is cheap to create, copy and pass by value, because it does not actually own the storage that it points to.

Performance Numbers: Sync (Single-Issue Sequential)

```
id,symbol,count,time
1,AAPL,565449,1.59043
2,AXP,731366,3.43745
3,BA,867366,5.40218
4,CAT,830327,7.08103
5,CSCO,400440,8.49192
6,CVX,687198,9.98761
7,DD,910932,12.2254
8,DIS,910430,14.058
9,GE,871676,15.8333
10,GS,280604,17.059
11,HD,556611,18.2738
12,IBM,860071,20.3876
13,INTC,559127,21.9856
14,JNJ,724724,25.5534
15,JPM,500473,26.576
16,KO,864903,28.5405
17,MCD,717021,30.087
18,MMM,698996,31.749
19,MRK,733948,33.2642
20,MSFT,475451,34.3134
21,NKE,556344,36.4545
```

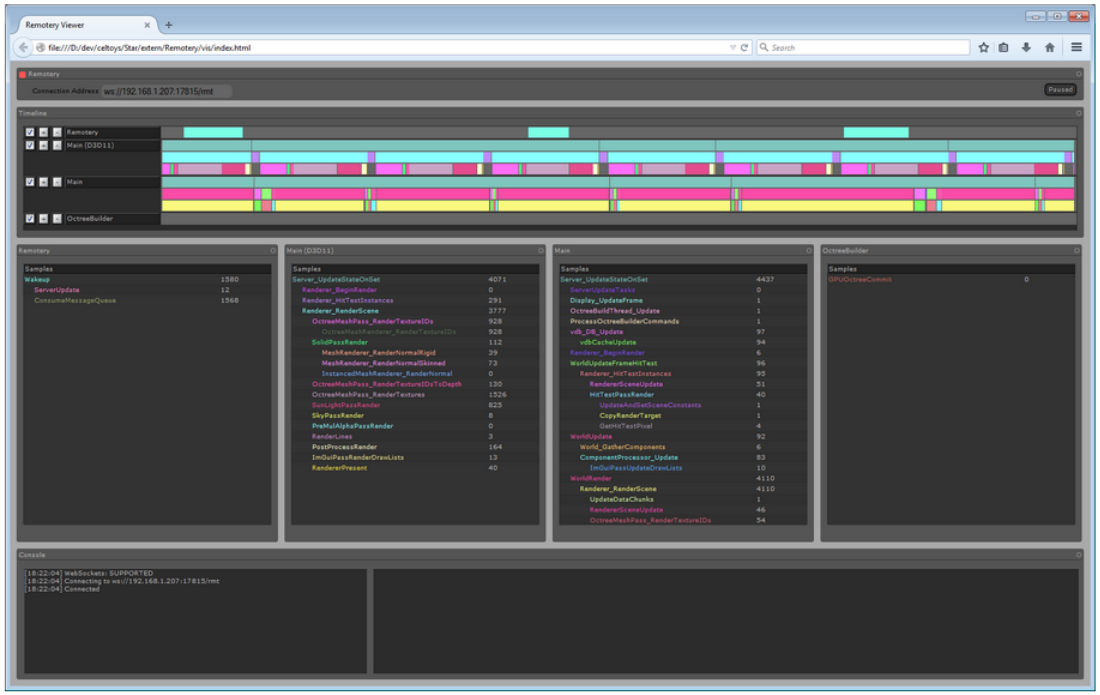
Performance Numbers: Async Pipeline

```
id,symbol,count,time
1,AAPL,565449,2.00713
2,AXP,731366,2.09158
3,BA,867366,2.13468
4,CAT,830327,2.19194
5,CSCO,400440,2.19197
6,CVX,687198,2.19198
7,DD,910932,2.51895
8,DIS,910430,2.51898
9,GE,871676,2.51899
10,GS,280604,2.519
11,HD,556611,2.51901
12,IBM,860071,2.51902
13,INTC,559127,2.51902
14,JNJ,724724,2.51903
15,JPM,500473,2.51904
16,KO,864903,2.51905
17,MCD,717021,2.51906
18,MMM,698996,2.51907
19,MRK,733948,2.51908
20,MSFT,475451,2.51908
21,NKE,556344,2.51909
```

Testing Performance

https://github.com/Celtoys/Remotery

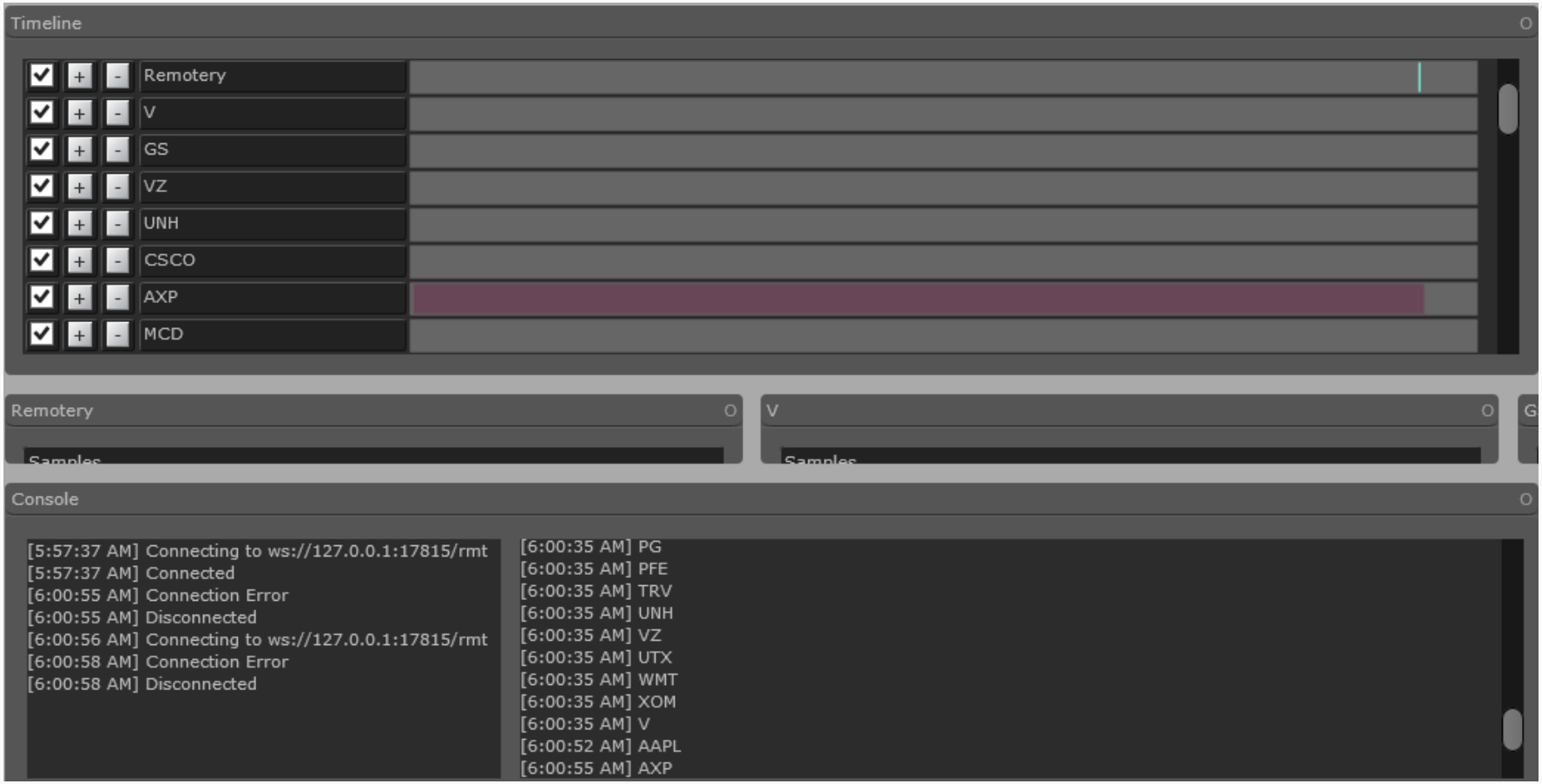
A realtime CPU/GPU profiler hosted in a single C file with a viewer that runs in a web browser.



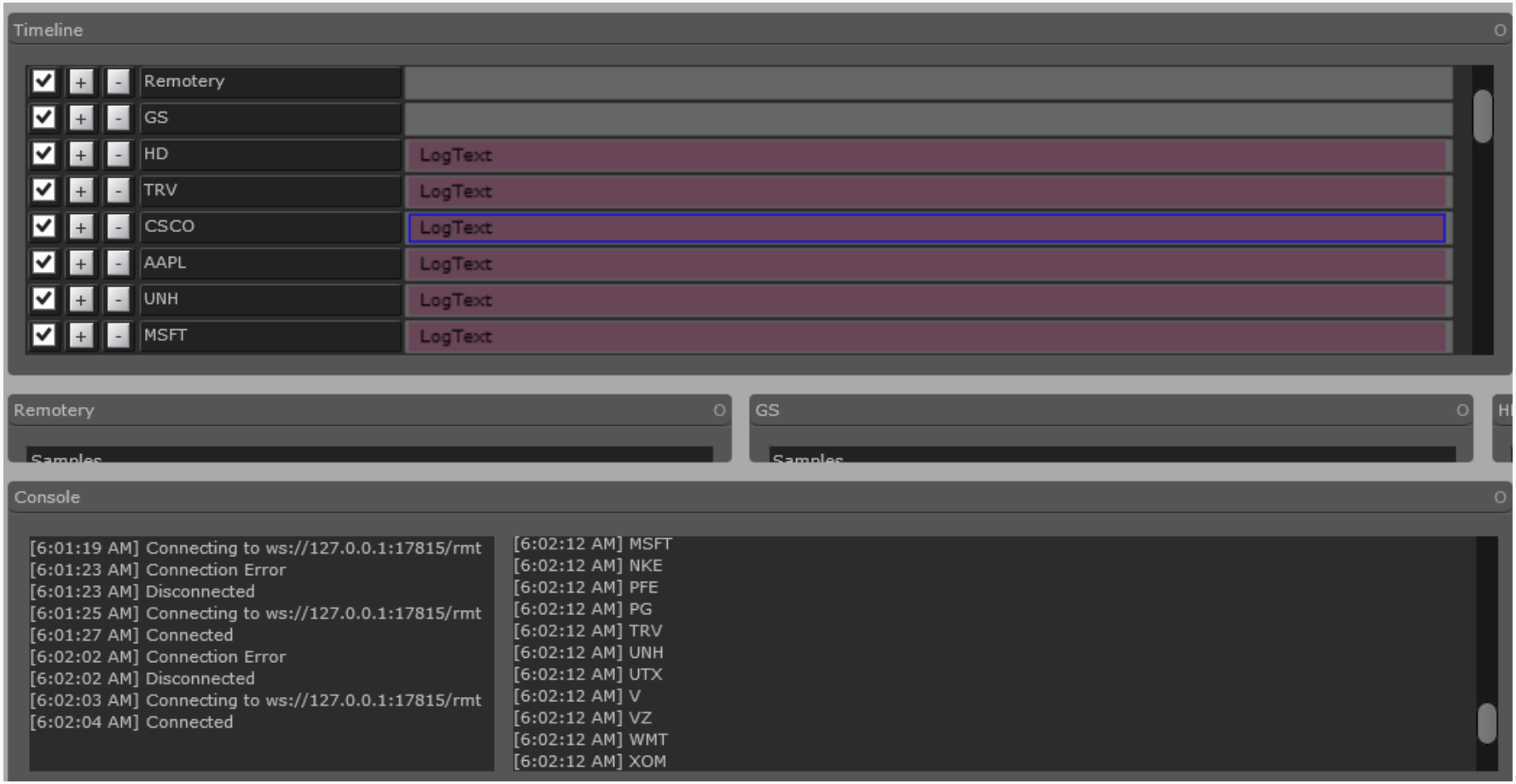
Supported features:

- Lightweight instrumentation of multiple threads running on the CPU.
- Web viewer that runs in Chrome, Firefox and Safari. Custom WebSockets server transmits sample data to the browser on a latent thread.
- Profiles itself and shows how it's performing in the viewer.
- Can optionally sample CUDA/D3D11 GPU activity.
- Console output for logging text.
- Console input for sending commands to your game.

Testing Performance: Without Pipelining



Testing Performance: With Pipelining



http://boost.org/libs/accumulators

Chapter 1. Boost.Accumulators

Eric Niebler

Preface

> "It is better to be approximately right than exactly wrong."
> -- *Old adage*

Description

Boost.Accumulators is both a library for incremental statistical computation as well as an extensible framework for incremental calculation in general. The library deals primarily with the concept of an *accumulator*, which is a primitive computational entity that accepts data one sample at a time and maintains some internal state.

http://theboostcpplibraries.com/boost.accumulators

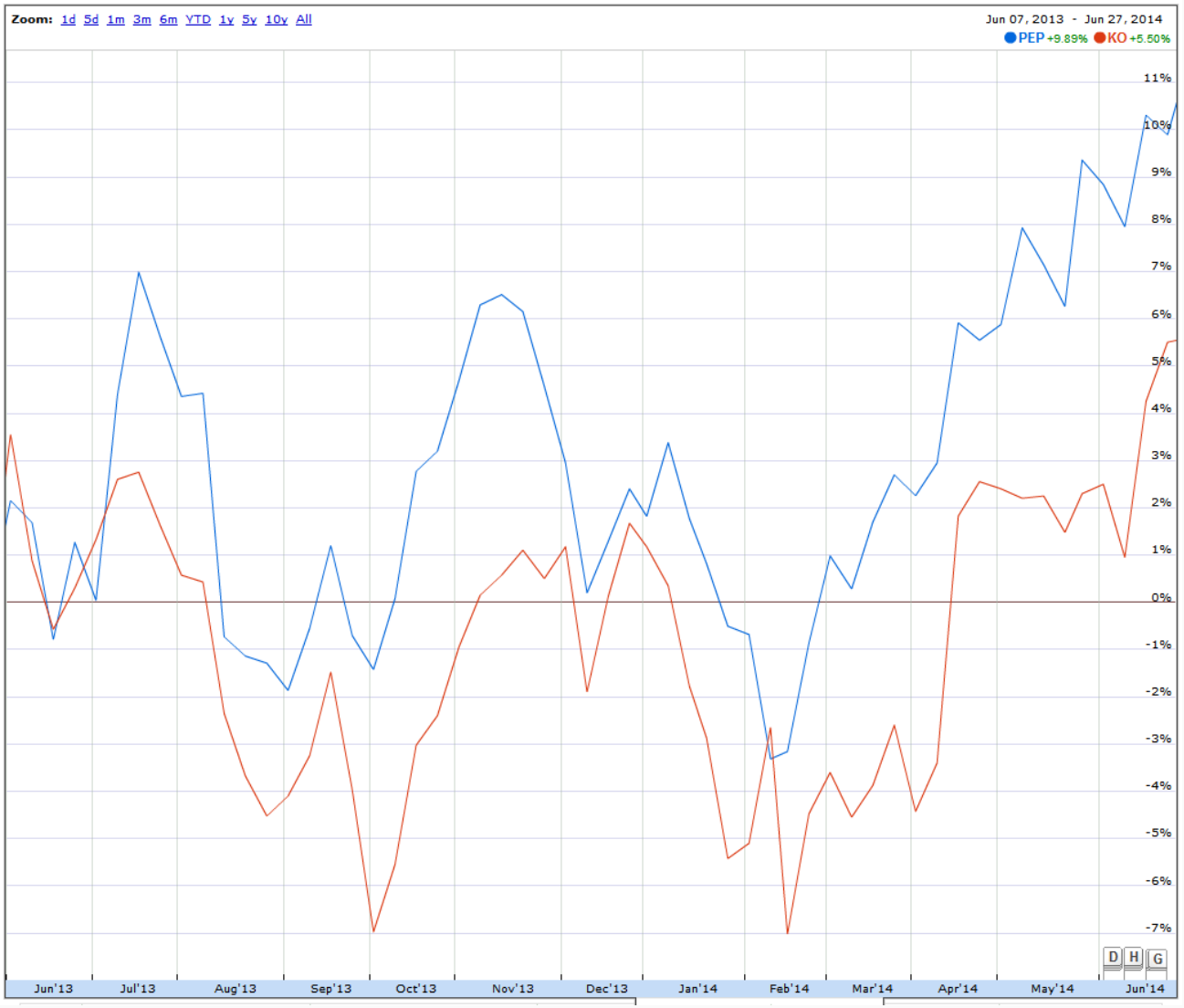
Boost.Accumulators

```
for (const auto & symbol : symbols)
{
  future_partial_results.push_back(std::async(std::launch::async, [&accumulators
  {
    auto & acc = accumulators[symbol];

    auto process_price = [&acc](double price)
    {
      acc(price);
    };
```

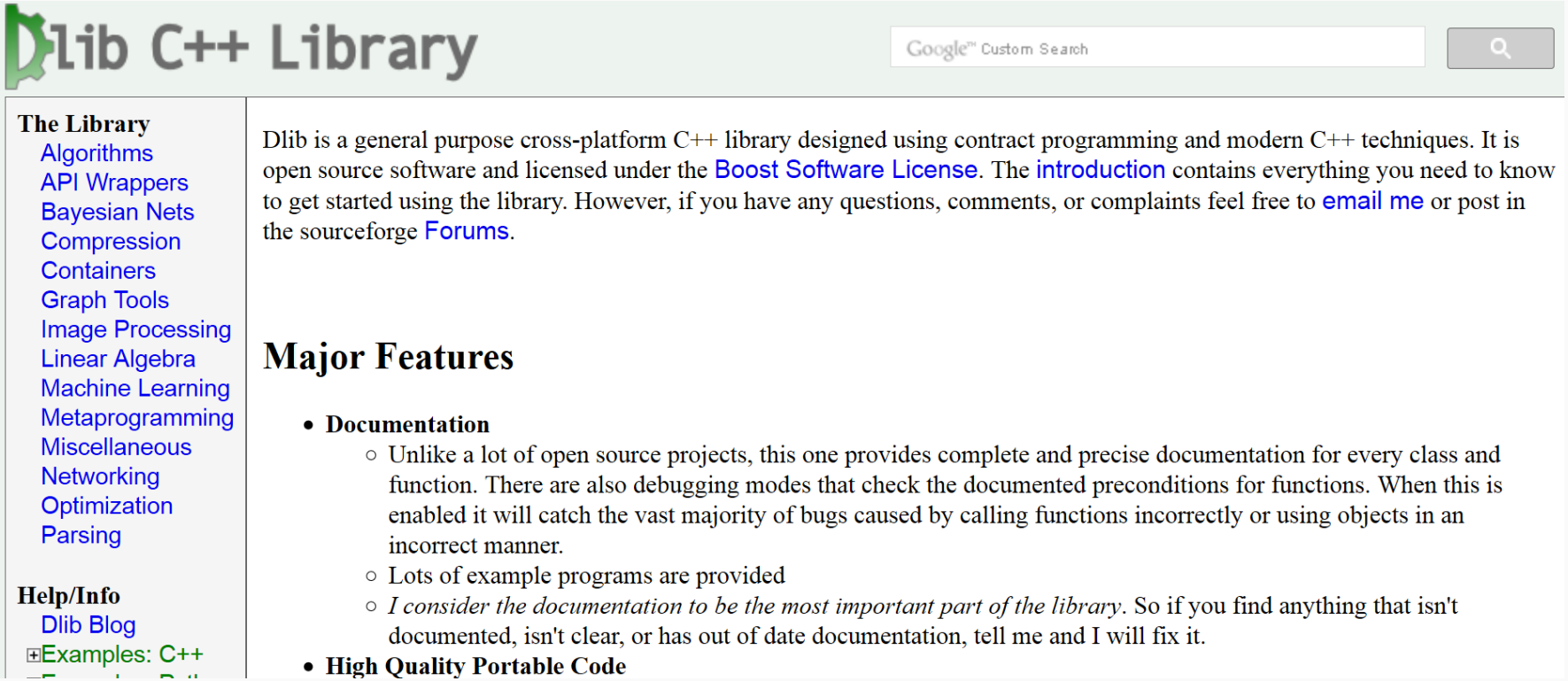
Boost.Accumulators

```
// perf overhead tests
auto do_nothing = [](double price){};

// use
namespace ba = boost::accumulators;
namespace bat = ba::tag;
//using accumulator_type = ba::accumulator_set<double, ba::features<bat::count>>;
using accumulator_type = ba::accumulator_set<double,
  ba::stats<bat::mean, bat::median, bat::variance, bat::skewness, bat::kurtosis>>;
```

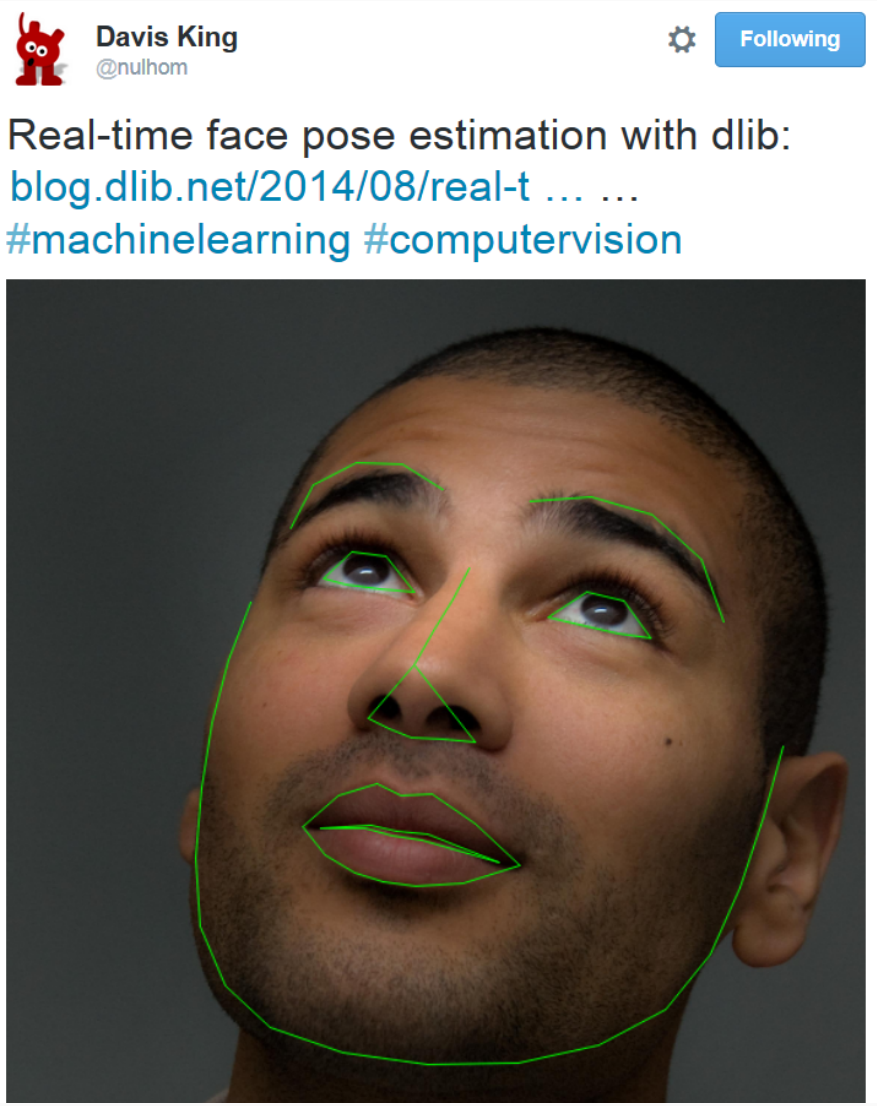
Incremental -> Machine Learning



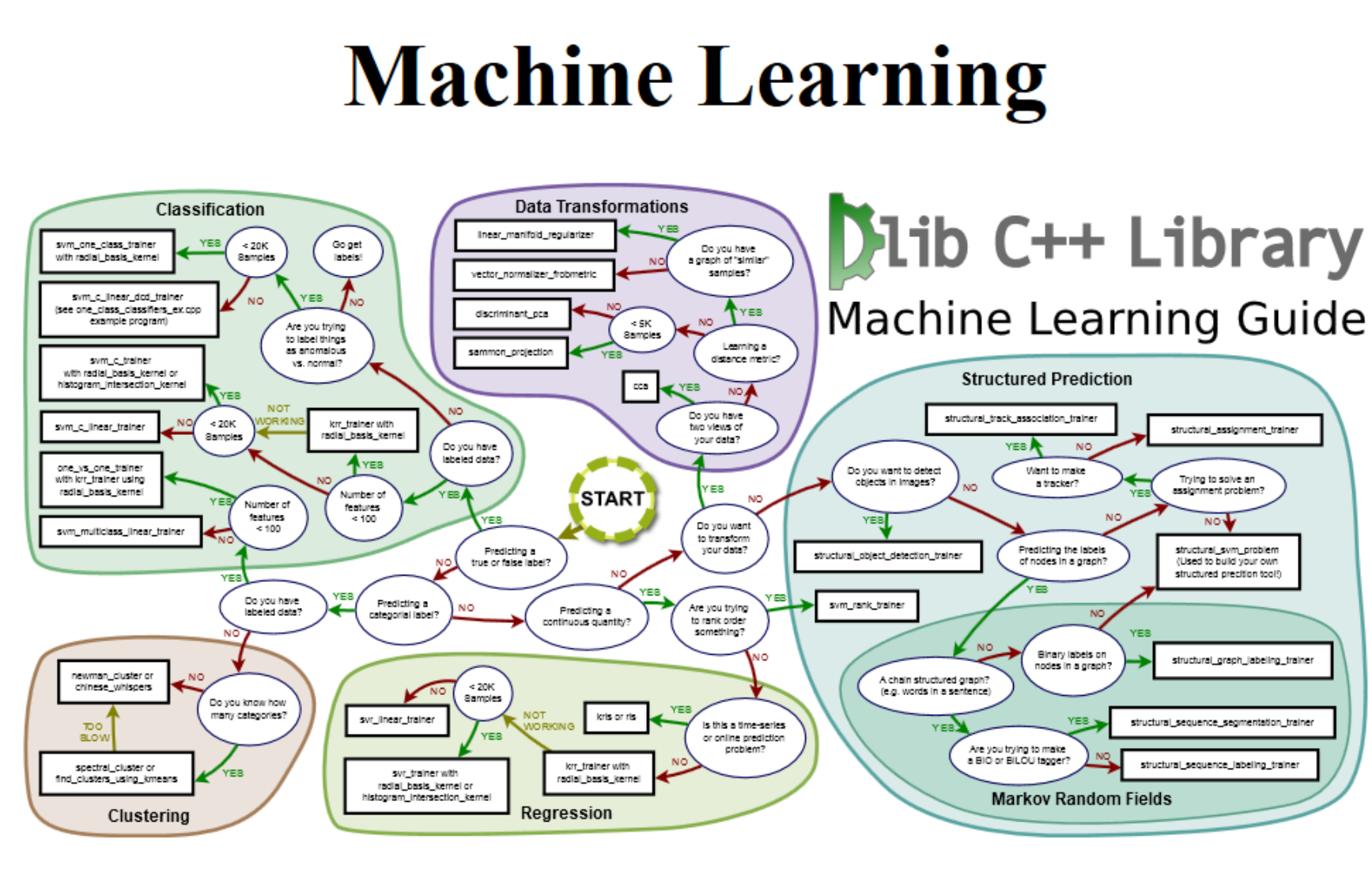
(On-Line) Machine Learning



(On-Line) Machine Learning



http://dlib.net/ml.html



DLIB - EXAMPLE

http://dlib.net/kcentroid_ex.cpp.html

```
This is an example illustrating the use of the kcentroid object
from the dlib C++ Library.

The kcentroid object is an implementation of an algorithm that recursively
computes the centroid (i.e. average) of a set of points.  The interesting
thing about dlib::kcentroid is that it does so in a kernel induced feature
space.  This means that you can use it as a non-linear one-class classifier.
So you might use it to perform online novelty detection (although, it has
other uses, see the svm_pegasos or kkmeans examples for example).
```

DLIB - EXAMPLE

```
using kernel_type = dlib::radial_basis_kernel<sample_type>;
using learner_type = dlib::kcentroid<kernel_type>;
learner_type  learner(kernel_type(0.1), 0.01, 15);
```

```
auto & learner = learners[symbol];

auto process_price = [&acc, &learner](double price)
{
  acc(price);
  sample_type sample;
  sample(0) = price;
  learner.train(sample);
};
```

Is all of this of any use?

?

Disclaimer: http://xkcd.com/1570/

Skill or luck?

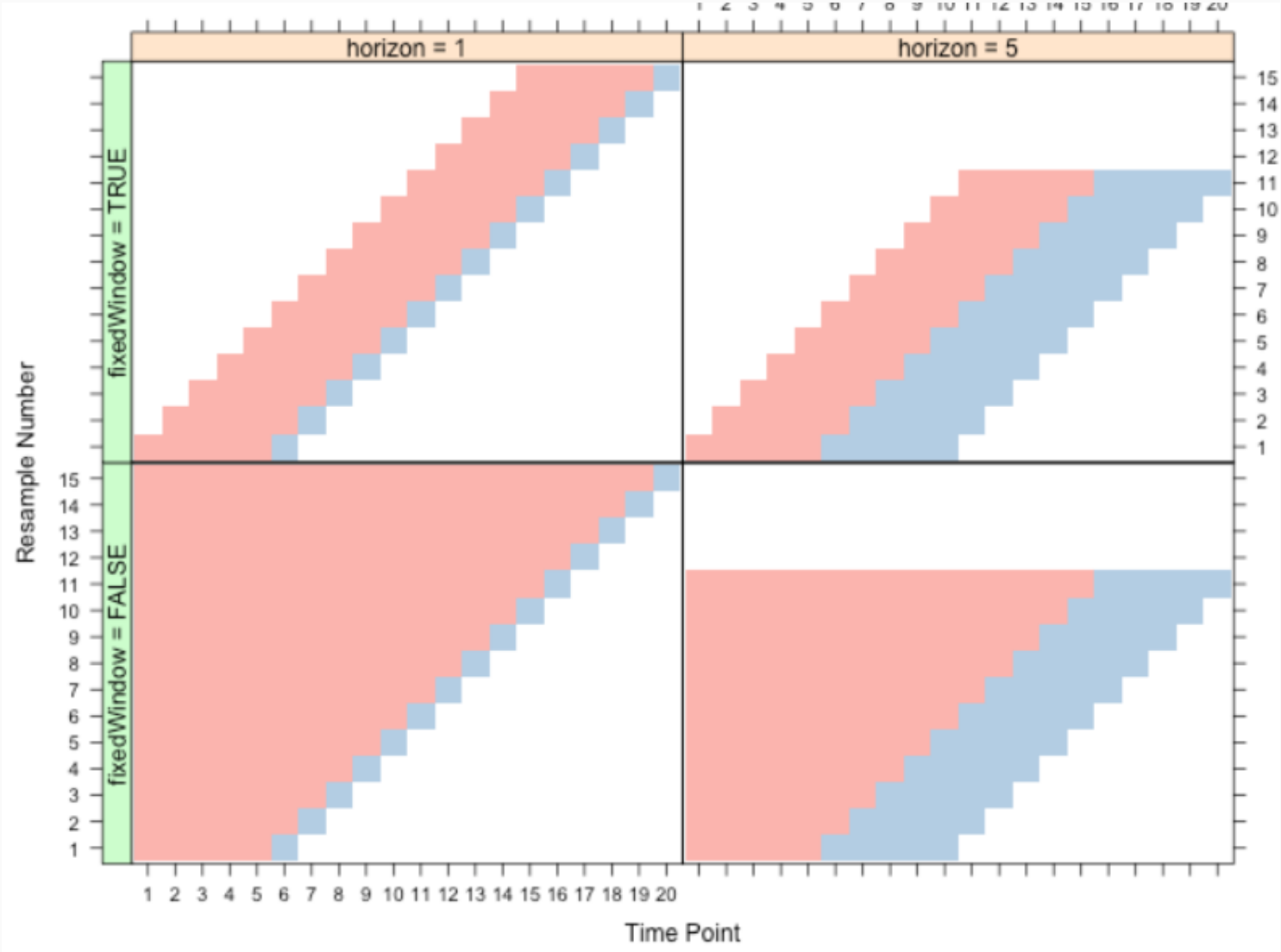
Forecasting: principles and practice

https://www.otexts.org/book/fpp

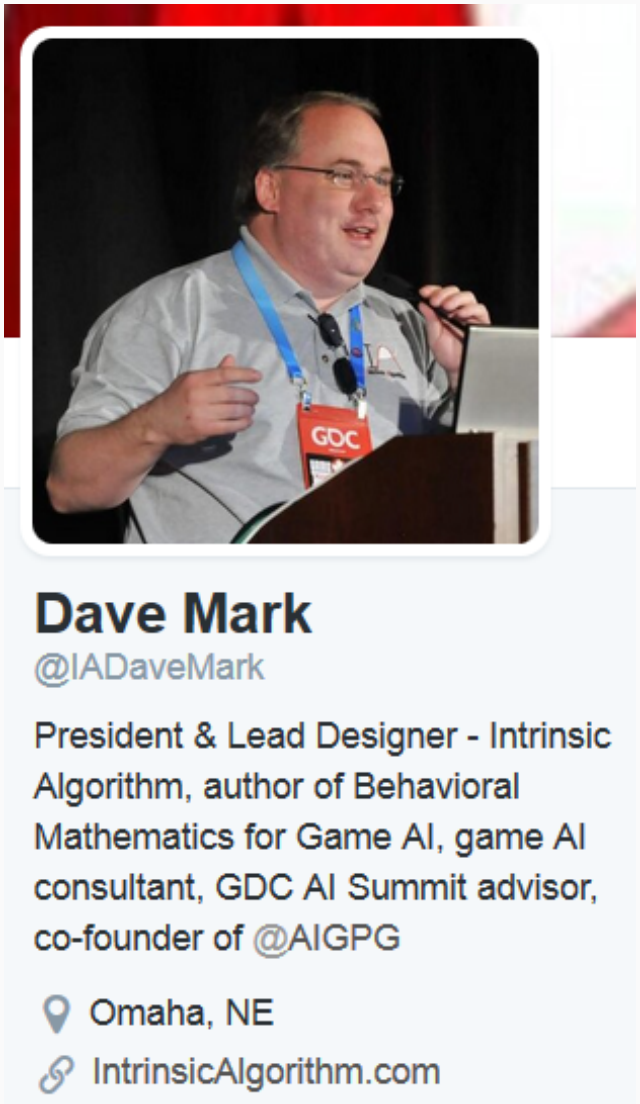
Skill or luck?

Evaluating forecast accuracy: https://www.otexts.org/fpp/2/5

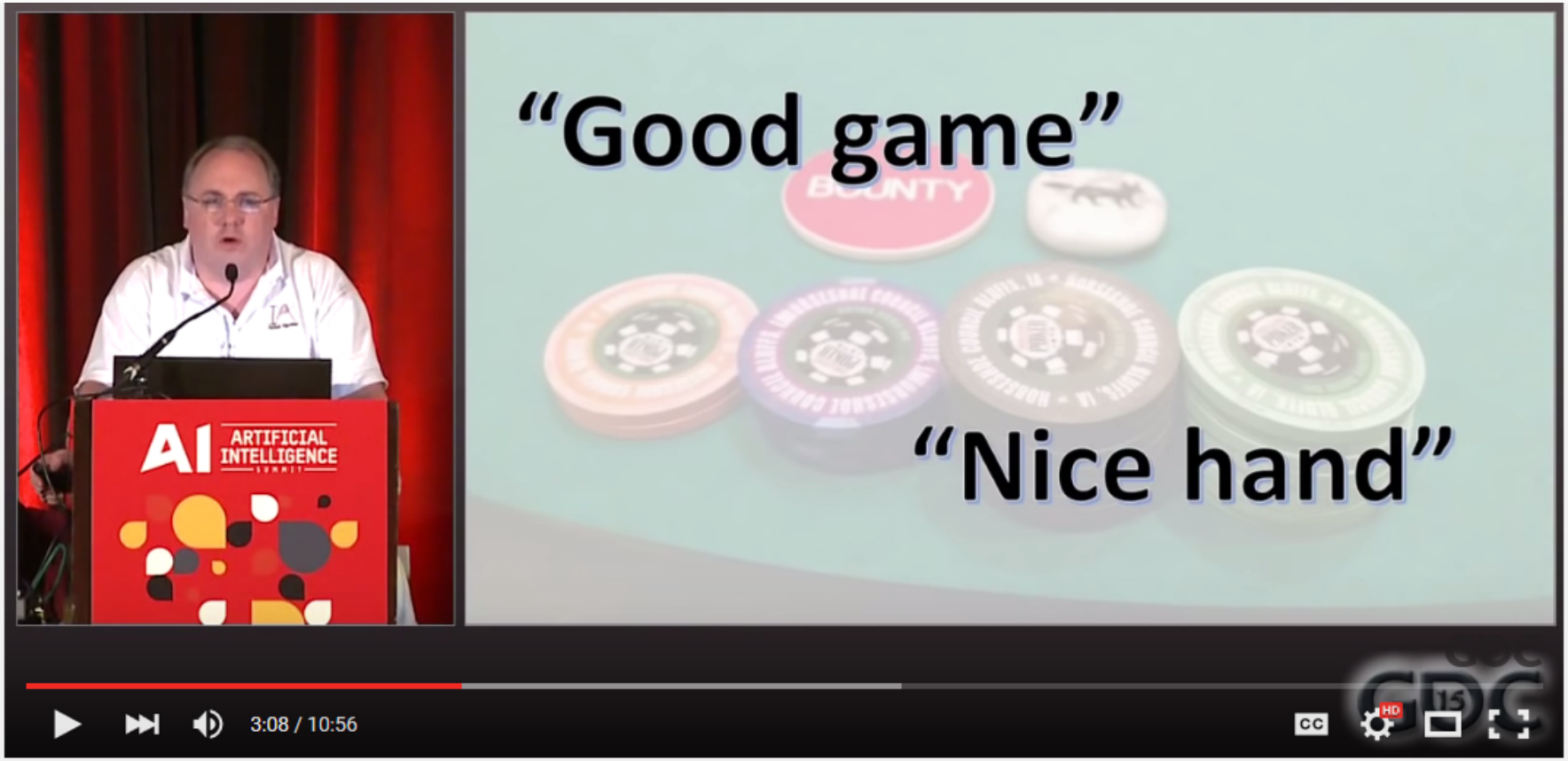
Data splitting: http://topepo.github.io/caret/splitting.html#time



Skill or luck?

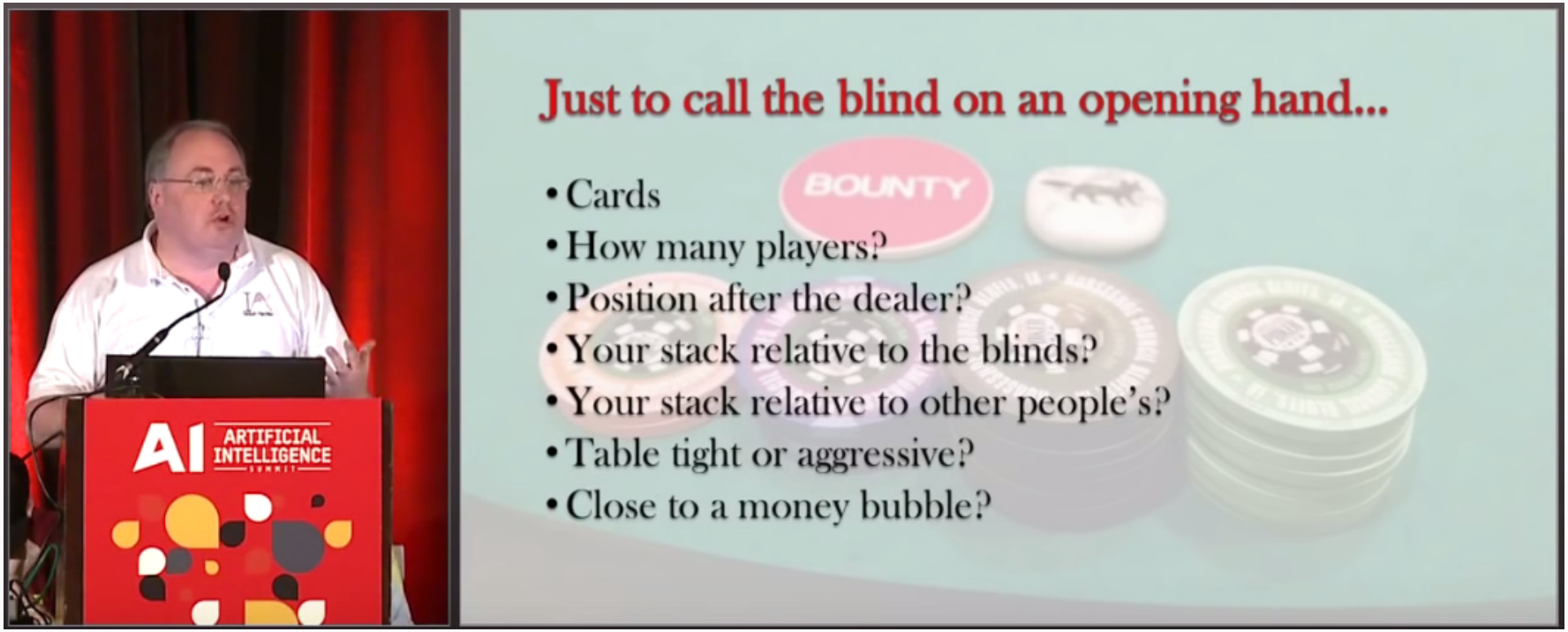


Dave Mark: Turing Tantrums, AI Summit @ GDC 2015



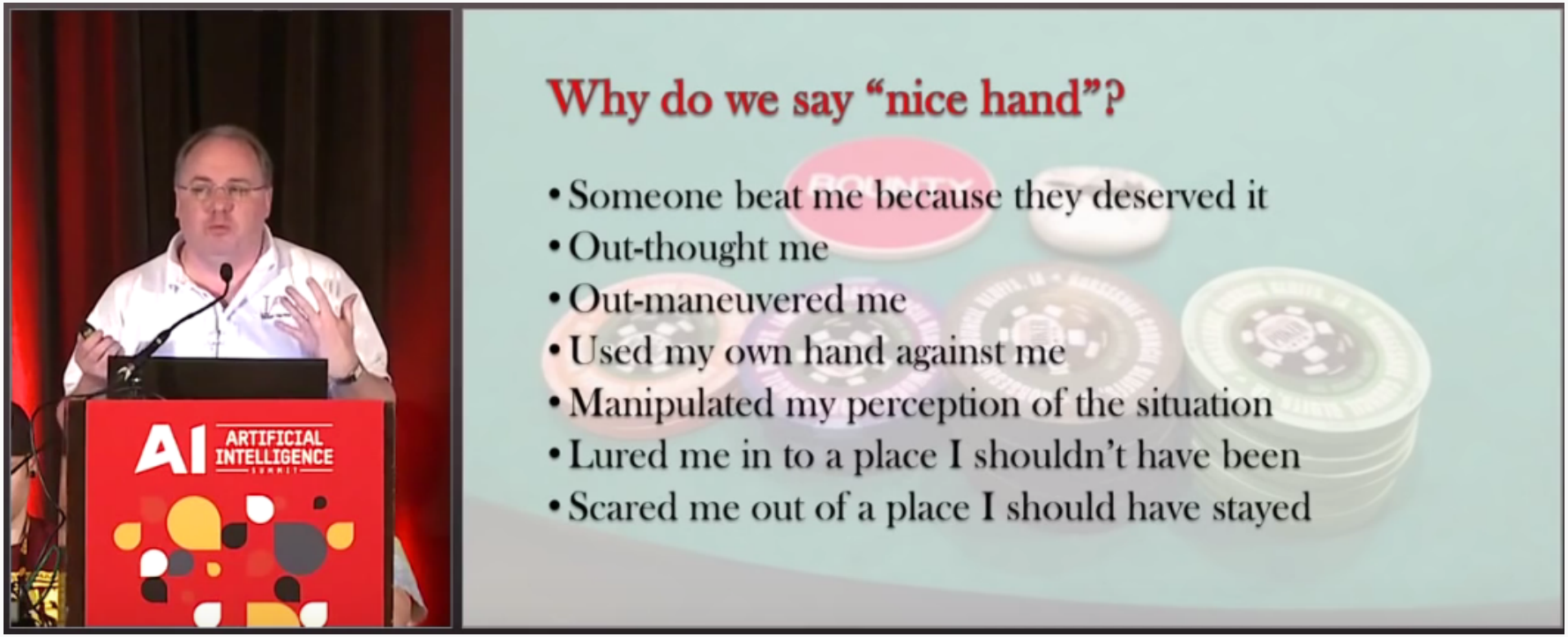
https://youtu.be/ei0V4AnbtNA

Dave Mark: Turing Tantrums, AI Summit @ GDC 2015



https://youtu.be/ei0V4AnbtNA

Dave Mark: Turing Tantrums, AI Summit @ GDC 2015



https://youtu.be/ei0V4AnbtNA

Testing

- Testing
  - Phil Nash: Test Driven C++ With Catch
  - http://www.levelofindirection.com/journal/2015/7/8/a-game-of-tag.html
  - https://www.snellman.net/blog/archive/2015-07-09-unit-testing-a-tcp-stack/

- Continuous Performance Management
  - Martin Thompson: ”Designing for Performance” https://youtube.com/watch?v=fDGWWpHlzvw
    - ”Performance test as part of Continuous Integration”
    - ”Can your acceptance tests run as performance tests?”
    - ”Build telemetry into production systems”
  - CPM for C++ http://baptiste-wicht.com/posts/2015/06/continuous-performance-management-with-cpm-for-cpp.html
  - Baseline(s) for CPM
    - Measure baseline overhead: NOP
    - Re-measure added overhead incrementally

```
auto do_nothing = [](double price) {};
auto process_price = do_nothing;
```

Testing and Performance

- Bryce Adelstein-Lelbach: Benchmarking C++ Code
  - Repeat tests: *Uncertainty!*
- Chandler: Tuning C++: Benchmarks, and Compilers, and CPUs! Oh My!
  - `perf` - *More than just counters*: https://perf.wiki.kernel.org/ (e.g., asm branches visualization)
  - more tools:
    - Linux Performance: http://www.brendangregg.com/linuxperf.html
    - `gcc-explorer`: https://github.com/mattgodbolt/gcc-explorer (e.g., asm <-> C++ code matching colorization)
- Rx(Cpp) for backtesting timing w/ virtual time:
  - https://github.com/Reactive-Extensions/RxCpp
  - `http://weareadaptive.com/blog/2015/07/16/historical-time-series-data-rx/`
  - http://blogs.msdn.com/b/rxteam/archive/2012/06/14/testing-rx-queries-using-virtual-time-scheduling.aspx

Performance and Latency

Standard Deviation and application latency should never show up on the same page

If you haven't stated percentiles and a Max, you haven't specified your requirements

Measuring throughput without latency behavior is [usually] meaningless

- http://www.azulsystems.com/presentations/qcon-ny-2015-how-not-to-measure-latency
- http://www.azulsystems.com/presentations/qcon-london-2014-understanding-latency
- http://psy-lob-saw.blogspot.com/2015/02/hdrhistogram-better-latency-capture.html

Communication

- Lock-free
  - e.g., http://moodycamel.com/blog
  - Preferable over MPMC: MPSC
- Fedor: Live lock-free or deadlock (practical Lock-free programming)
- Pedro: How to make your data structures wait-free for reads
  - bounded tail-latency (readers)
  - http://concurrencyfreaks.blogspot.com/
- Michael: C++11/14/17 Atomics the Deep dive: the gory details, before the story consumes you!

Communication

- Artur: Concurrency TS: The Editor's Report

  http://www.boost.org/doc/libs/release/doc/html/thread/synchronization.html
- Pablo: Parallel Program Execution using Work Stealing
- Gor: C++ Coroutines - a negative overhead abstraction
  - http://wg21.link/n4134 - Asynchronous I/O, tcp_reader
- Paul: C++ Atomics: The Sad Story of memory_order_consume: A Happy Ending at Last?
  - Is Parallel Programming Hard, And, If So, What Can You Do About It?:

    https://www.kernel.org/pub/linux/kernel/people/paulmck/perfbook/perfbook.
  - What is RCU, Fundamentally?:
    https://lwn.net/Articles/262464/
- WIP, **Feedback Wanted & Welcome!**:
  https://github.com/MattPD/cpplinks/

  atomics.lockfree.memory_model.md

Physical Models: 1 nanosecond

1 ns

The Queen of Code — http://tnw.me/7y5IzvF

Physical Models: 1 microsecond

1 μs

The Queen of Code — http://tnw.me/7y5IzvF

Lecture 7. Pipelining - Carnegie Mellon - Computer Architecture 2015 - Onur Mutlu
https://youtu.be/dKXbONPqBNY

Remember: An Ideal Pipeline

- Goal: Increase throughput with little increase in cost (hardware cost, in case of instruction processing)
- Repetition of identical operations
  - The same operation is repeated on a large number of different inputs (e.g., all laundry loads go through the same steps)
- Repetition of independent operations
  - No dependencies between repeated operations
- Uniformly partitionable suboperations
  - Processing an be evenly divided into uniform-latency suboperations (that do not share resources)
- Fitting examples: automobile assembly line, doing laundry
  - What about the instruction processing "cycle"?

https://youtu.be/dKXbONPqBNY

Instruction Pipeline: Not An Ideal Pipeline

- Identical operations ... NOT!
  - ⇒ different instructions → not all need the same stages
    Forcing different instructions to go through the same pipe stages
    → external fragmentation (some pipe stages idle for some instructions)
- Uniform suboperations ... NOT!
  - ⇒ different pipeline stages → not the same latency
    Need to force each stage to be controlled by the same clock
    → internal fragmentation (some pipe stages are too fast but all take the same clock cycle time)
- Independent operations ... NOT!
  - ⇒ instructions are not independent of each other
    Need to detect and resolve inter-instruction dependencies to ensure the pipeline provides correct results
    → pipeline stalls (pipeline is not always moving)

https://youtu.be/dKXbONPqBNY

Slides

https://speakerdeck.com/mattpd

Thank You!
Questions?