Browsing Web 3.0 on 3.0 Watts

or Why Browsers Will Be Parallel and Implications for Education

Rastislav Bodik

with Chris Jones, Rose Liu, Leo Meyerovich,
Krste Asanovic and the rest of Par Lab

UC Berkeley

Milestones in computing?

What are your top 3 milestones in computing?

- enabled new ways of thinking/doing things

Some candidates:

- hard disks and databases
- languages and their compilers
- the transistor
- theory of NP-completeness
- Web/internet
- the PC

Bell’s Law of Computer Classes

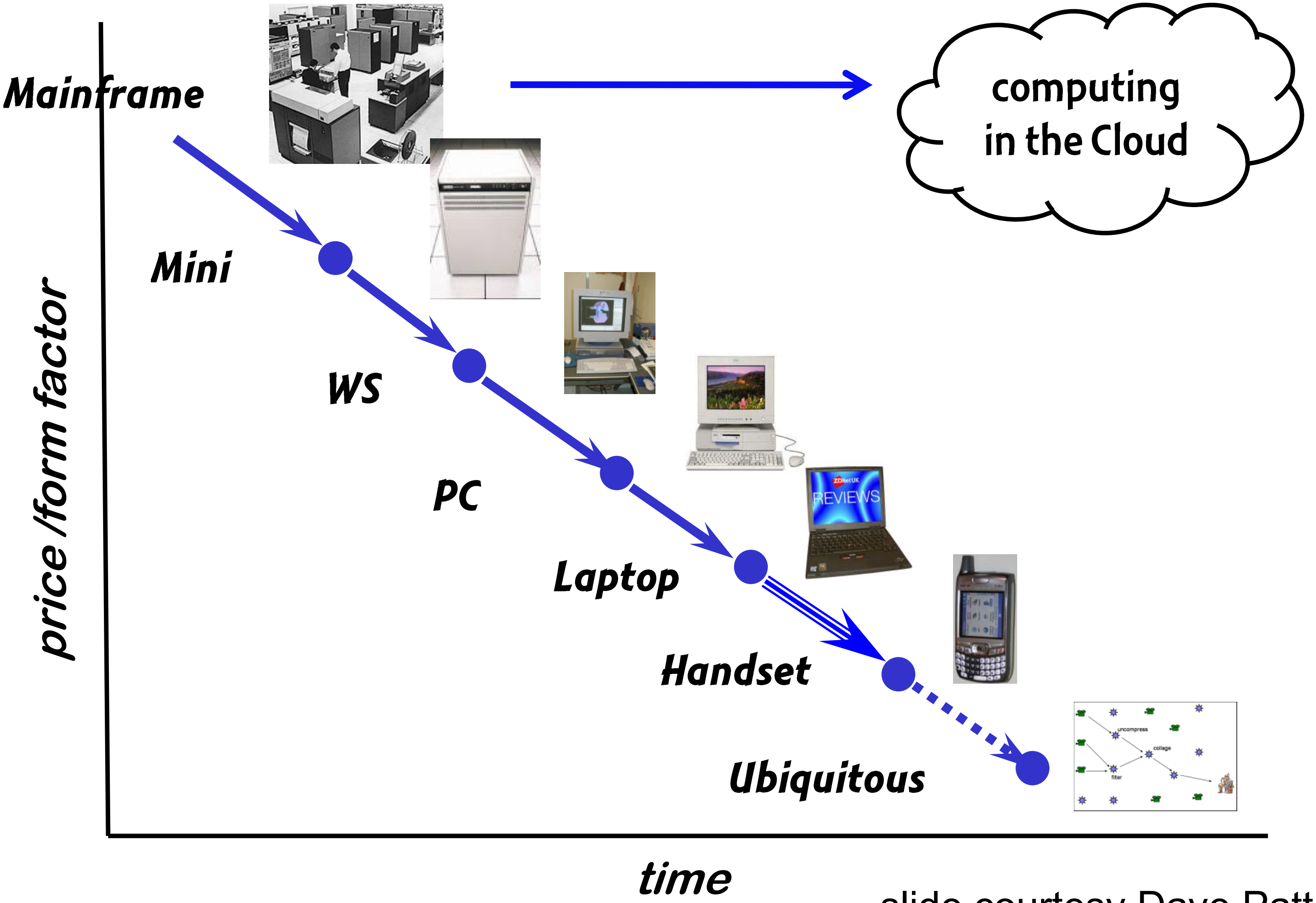
Moore's Law Enables Two Evolution Paths

Path 1: increasing performance, same cost and form factor

Path 2: constant performance, decreasing cost and form factor

slide courtesy Dave Patterson

Bell's Law: a Corollary of Moore's Law



Mainframe

Mini

WS

PC

Laptop

Handset

Ubiquitous

computing in the Cloud

price /form factor

time

slide courtesy Dave Patterson

Handheld's Killer Apps

The laptop→handheld transition will happen if ...

- handhelds do to laptops what laptops did to desktops

Handheld killer apps that will make laptops irrelevant:

1. Device convergence
   - phone + music player + PDA + gaming
2. Laptop replacement
   - small form factor = convenience
3. Personal assistant
   - vision, hearing, memory, health improvement

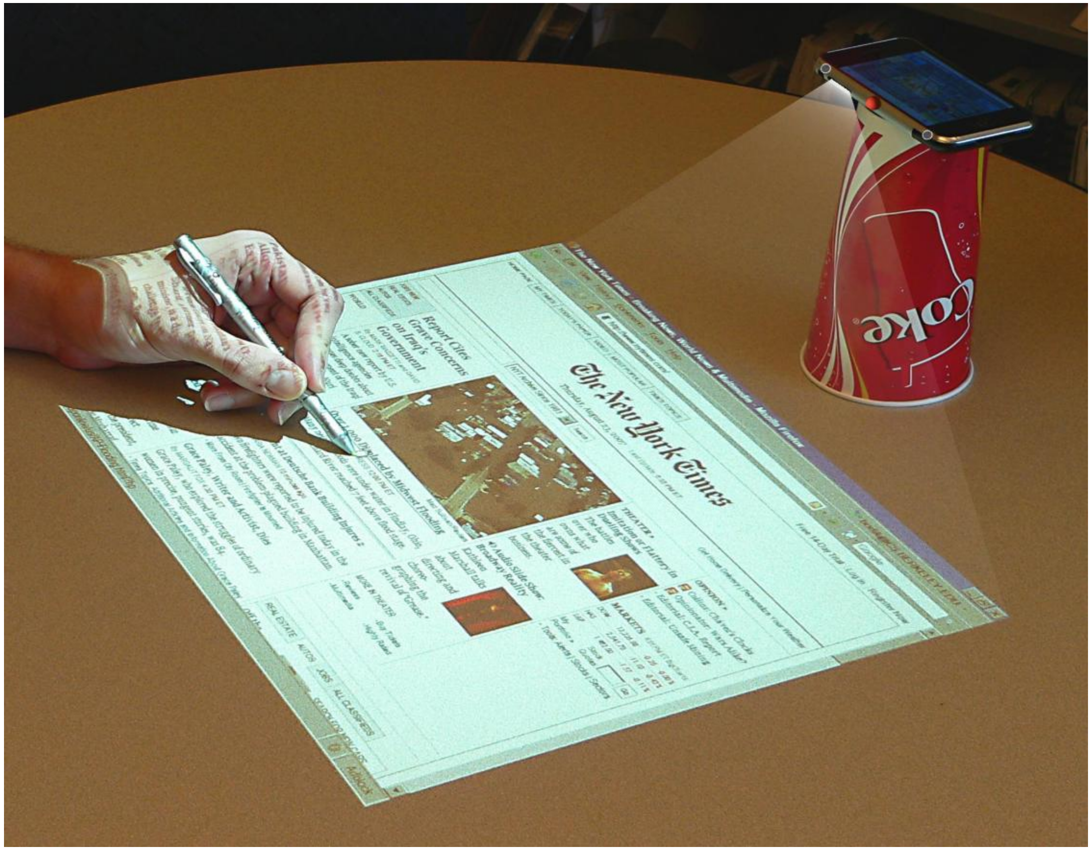
Handheld as a Laptop Replacement

- When will handhelds take over?
  - when they have laptop-quality browser

- Why is a web browser enough?
  - **In Web 3.0:** most apps will be browser-based
  - **In Web 2.0:** some apps are still native (Picassa, Google Earth)

A handheld browser may be soon possible

A guy walks into a bar, asks for a cup, and starts his browser.

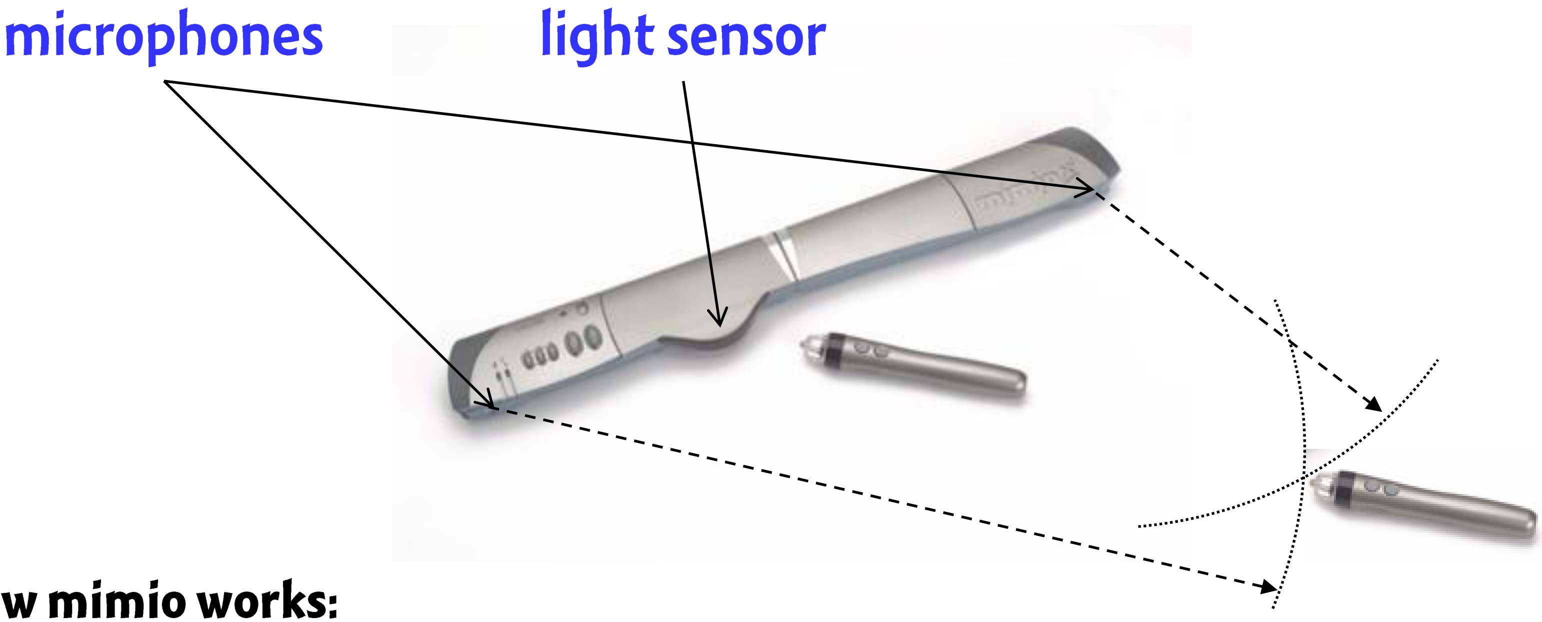


Display: cell phone projector or “wearable”

Texas Instruments, CES 2008

Input: idea for tablet input for a handheld

- **Inspiration:** mimio, a whiteboard recorder (mimio.com)



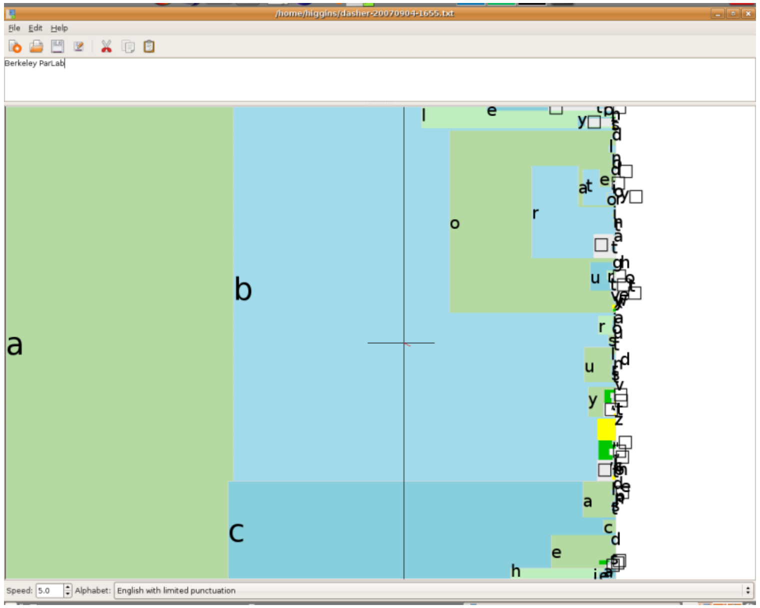
How mimio works:
triangulates in the same way that one measures lightning distance

1. marker simulates a lightning strike: simultaneously emits light and sound signals;
2. capture bar measures sound travel time: yields marker distance to each mic;
3. the two distances determine marker location on the whiteboard; goto step 1

Dasher + picomimio = keyboard-rate input

Dasher: replacement for traditional keyboards

- Input rates up to ~30 words/minute
- Only needs 1 input axis (up/down) to work
  - can be controlled by picomimio, eyes, tilt sensor, …


/home/higgins/dasher-20070904-1655.txt

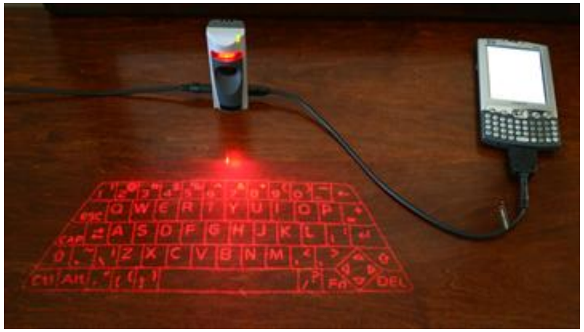
Berkeley ParLab

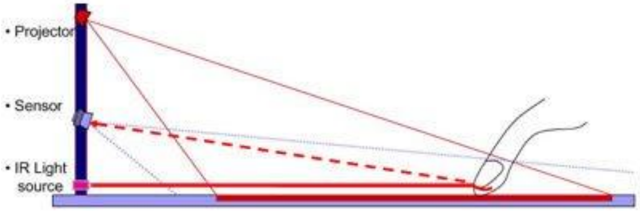
Speed: 5.0 Alphabet: English with limited punctuation

See http://www.inference.phy.cam.ac.uk/dasher/ for more info, online demo

Other input alternatives

- speech recognition
  - we'll have enough MIPS for low-noise environments
- sensors for gestures
  - iPhone has sensors: ex. to zoom into a map, move it so
- virtual laser keyboard:





What about CPU performance?

Display: many alternatives

Input: half as many

Network: plenty fast soon (all we need is better providers)

CPU speed: no longer considered a bottleneck. Is it true?

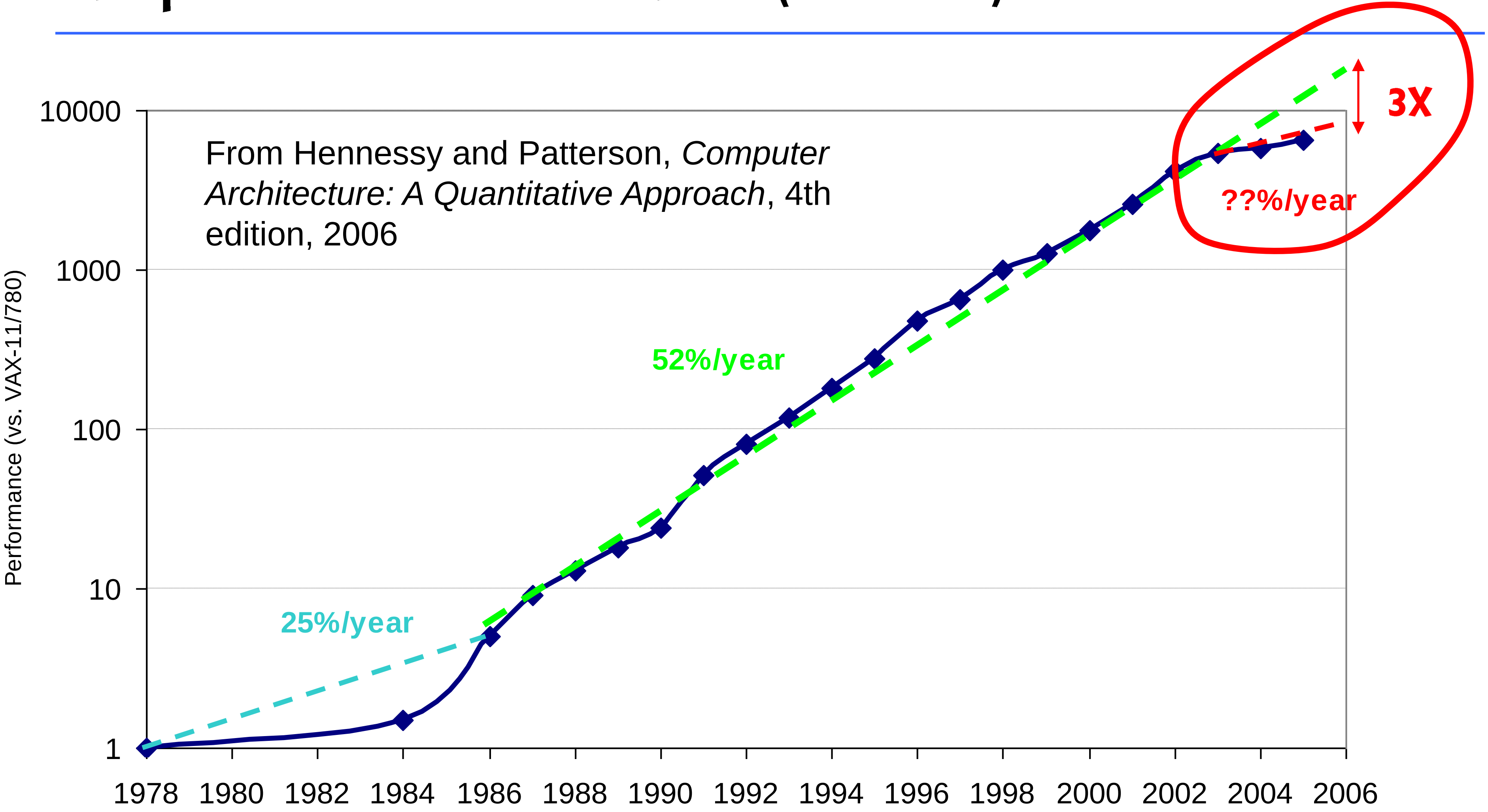
Why is iPhone slow? Network or CPU?

Experiment: time to load+render cnn.com on two networks

machine	*seconds*
a modern desktop (2Mbps network)	2
T40 1.6GHz (a very old laptop; 2Mbps network)	7
T40 1.6Ghz (same laptop&network, battery mode)	13
iPhone 600MHz (2Mbps network)	37
iPhone 600MHz (1Mbps network)	40

☞ CPU performance is critical

Uniprocessor Performance (SPECint)



From Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 4th edition, 2006

Proxy Web Servers?

Recently introduced Skyfire browser uses a proxy server

- server parses the web page, sends images to handheld
- not unlike the X11 client/server architecture

Limitations:

- network latency: impact on interactivity
- radio power: may be cheaper to decompress than to receive
- server CPU cycles needed: average browser CPU load is 5%!

End of Bell's Law?

Undergraduate Course in PL and Compilers

Is our field slowly dying?

Observations from graduate admissions:

- AI, bio, NLP, OS much better at recruiting the brightest minds

This may have roots in undergraduate education

Enrollments at Berkeley:

- Algo=165 ; OS=155; AI=160; DB=110
- compare with compilers=60

The trend goes beyond Berkeley, I suspect

- Amazon rank for Silberschatz =800
- for Dragon Book=8,000

Why this lack of interest?

So much excitement

- eg, Web languages and frameworks

So many interesting open problems remain

- eg, parallelism, speeding up scripting languages

Recognition: 6 out of last 10 Turing Awards to our discipline

- perhaps this is a sign of "mature and solved"?

Is AI viewed as the hero of web2.0 success (web search)?

- while web2.0 is arguably a programming success

Education problem? Political mistakes? Marketing?

- No longer a required "core" course?

Relevance?

My survey why people take AI not compilers

- compilers "too hard", "solved, old stuff"

OS texbook revised more often

- Silberschatz in 7th edition
- Dragon in 2nd edition

I looked at a few OS lectures

- was stuff I wanted to learn! (eg PKK, peer-to-peer networks)
- compare this with register allocation

End of Bell's Law?

↓

Course in PL and Compilers

Browsers for Handhelds

Good news, bad news

Current laptops: how powerful?

~20GOPS at 20W, more with SIMD

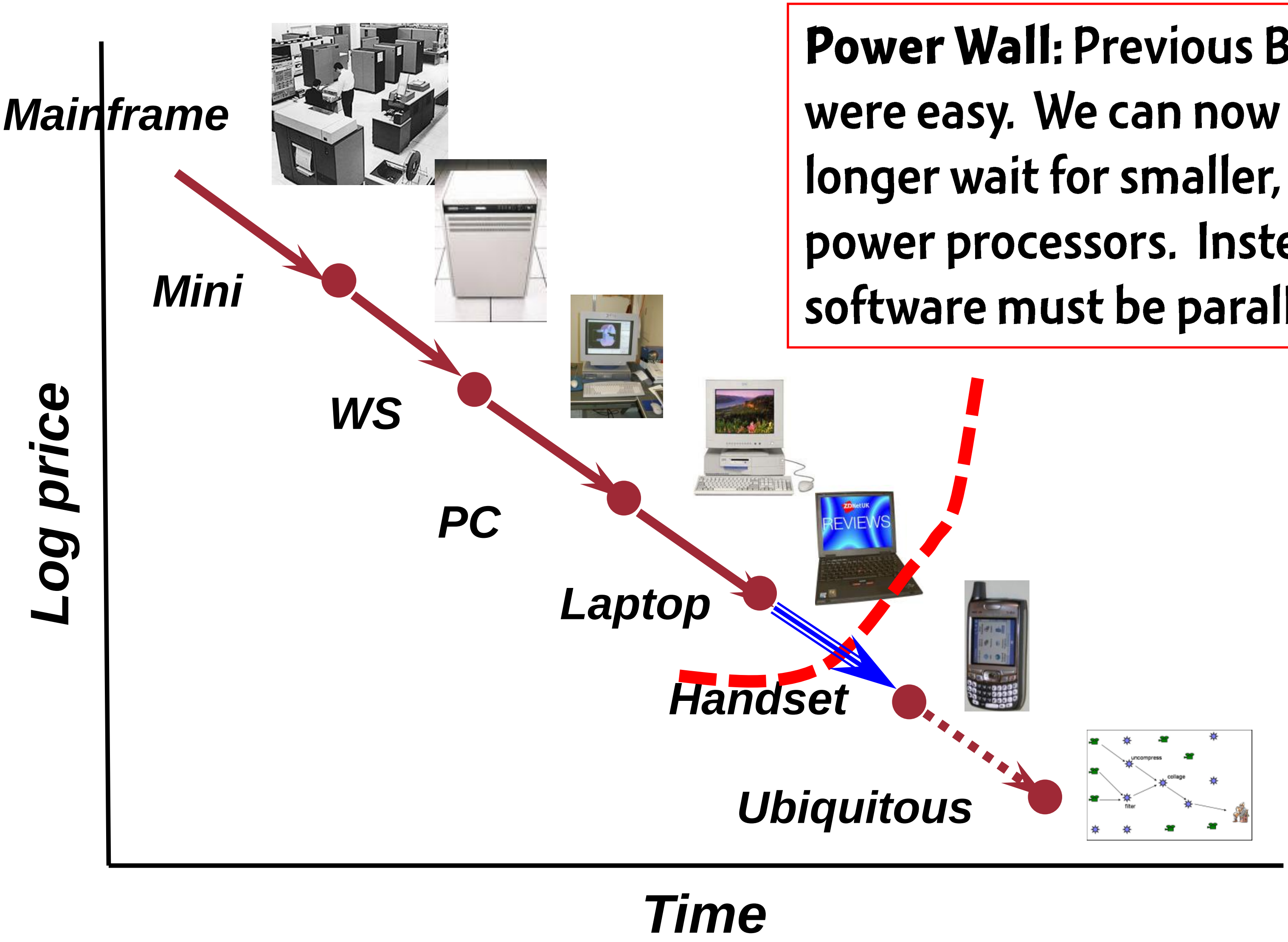
Good news: we should be able to get 50GOPS at 2W

- even in current 65nm technology (40pJ/op)

Bad news: the 50GOPS will come from 10-100 parallel cores

☞☞☞ **We must build a parallel browser**

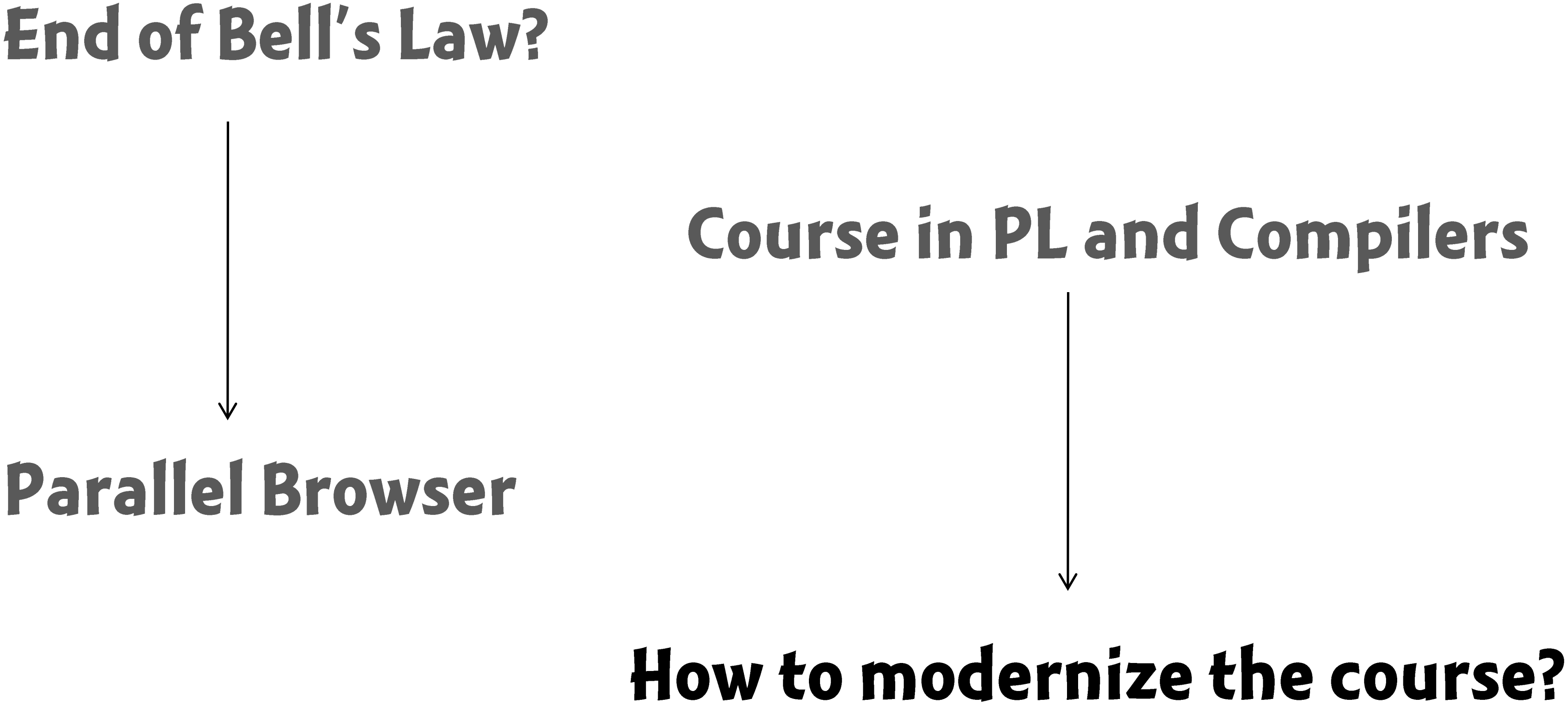
Transition to handhelds qualitatively different



Power Wall: Previous Bell steps were easy. We can now no longer wait for smaller, lower power processors. Instead, software must be parallelized.

Browser autopsy: three major components

- web page "compilation": HTML to DOM
  - lexing, parsing, and syntax-directed translation
  - **our project:** parallelize all three
- scripting
  - JavaScript (AJAX)
  - **our project:** a dataflow-ish web client language
- page layout and rendering:
  - similar to formatting a latex document
  - **our project:** design web language hand in hand with layout/rendering
- But these bottlenecks invoke Amdahl's Law
  - current browsers "inherently" sequential
  - for neither of these components, parallel successes exist



End of Bell's Law?

Parallel Browser

Course in PL and Compilers

How to modernize the course?

Experience from Berkeley

- My students and I have been remodeling the undergraduate PL and compiler course since 2003
- Two major revisions, each offered twice
- Student survey results on "usefulness of the course":

Fall 2003	5.4/7.0	
Fall 2004	5.8	
Spring 2007	5.9	
Fall 2007	6.3	← 15-year high for the course

No less "useful" than OS, AI. Still limping behind architecture.

How to revise course?

Our initial thinking in 2003: *Modernize it!*

- incorporate latest research: latest analyses, type systems

Then the key question arose: *Who is the target audience?*

- compiler writers or software engineers?

But do programmers need language technologies?

- surprisingly yes

But the content needs to go back to basics

Why do you want to take the course?

Write code that writes code.

- don't write "mindless" code, write a generator
- ex: database schema→ Java wrapper classes

New languages will keep coming. Be ready.

- CSS, HTML, JavaScript, JSP, PHP, Python, Ruby, XML
- know advantages of picking a particular language

Develop your own language.

- Many web languages created practically in the garage , not in a research lab: PHP, JavaScript, Ruby, perl

Learn about compilers and interpreters.

- language is the most important of all programmer tools
- Typical job ad: if you had to take one course, it'd be PL+compilers

Take cs164. Become unoffshorable.

"We design them here, but the labor is cheaper in Hell."

languages, frameworks, APIs

Change 1: Raise Level of Abstraction

What are the source/target programming model?

Before: compile Java-like → x86-like

Now: compile a Dataflow extension to AJAX → JavaScript

Change 2: Language landscape is broader today

- Dynamic languages in wide use
- Interpreters in wide use

- Programmer choice of data structures in dynamic , interpreted makes huge difference on performance

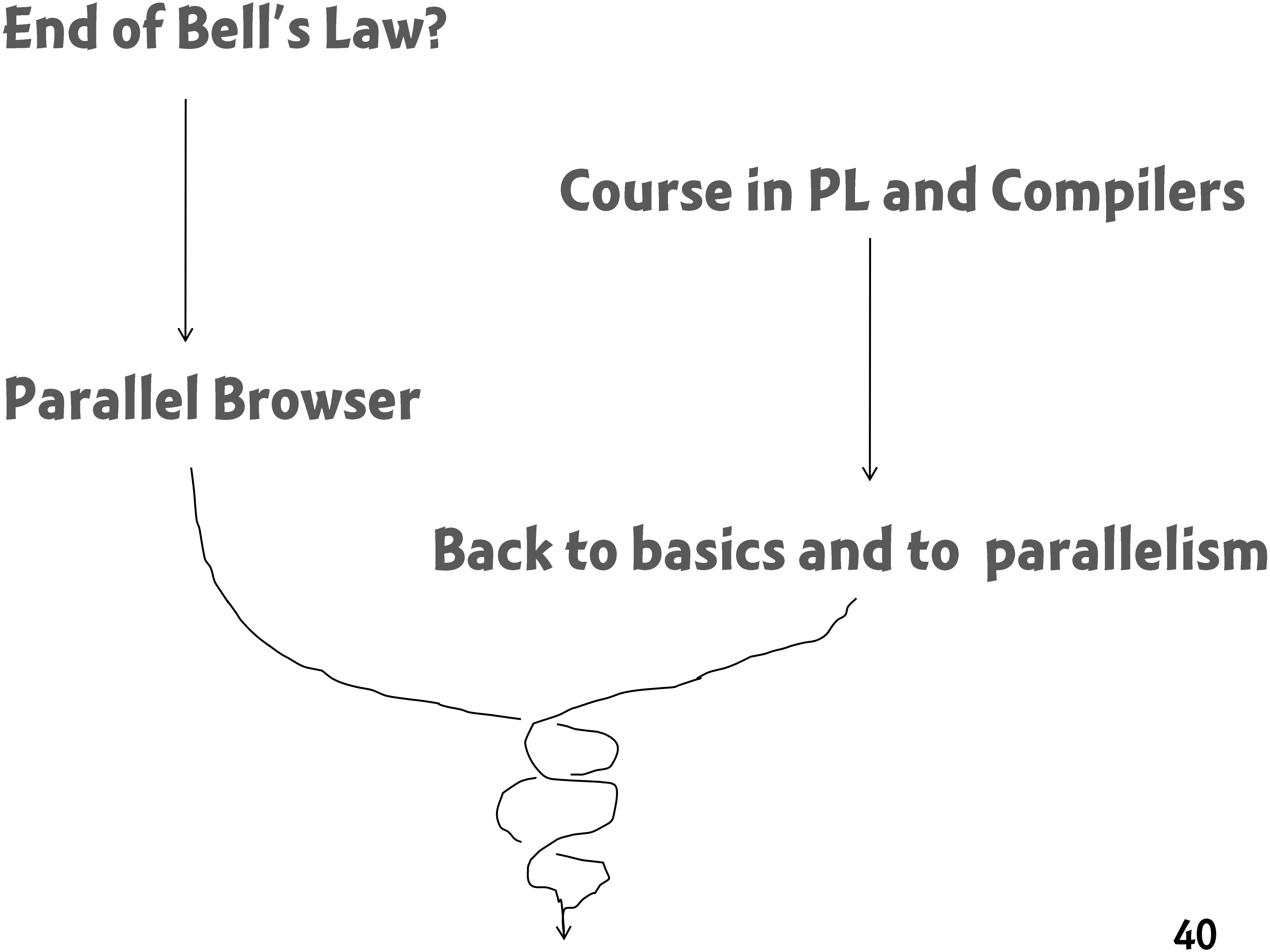
Change 3: Back to Basics

Some course content decisions go back 30 years

- when computers were 1,000,000-times slower
- when those topics were research topics

Go back to simple parsers: CYK and Earley

- CYK can be explained 10-times faster than LALR
- easy to implement → leads to better understanding

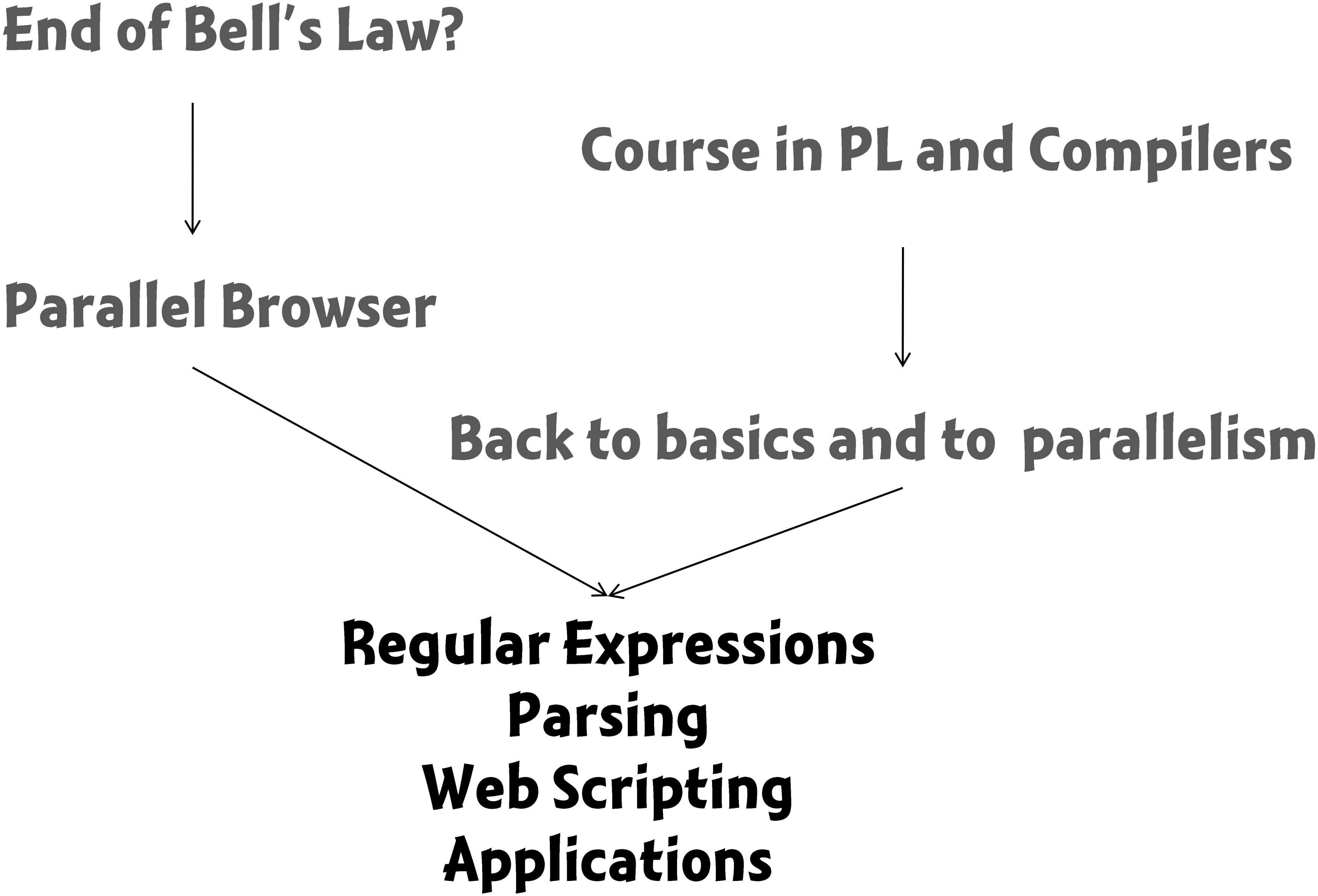


End of Bell's Law?

Parallel Browser

Course in PL and Compilers

Back to basics and to parallelism



End of Bell's Law?

Parallel Browser

Course in PL and Compilers

Back to basics and to parallelism

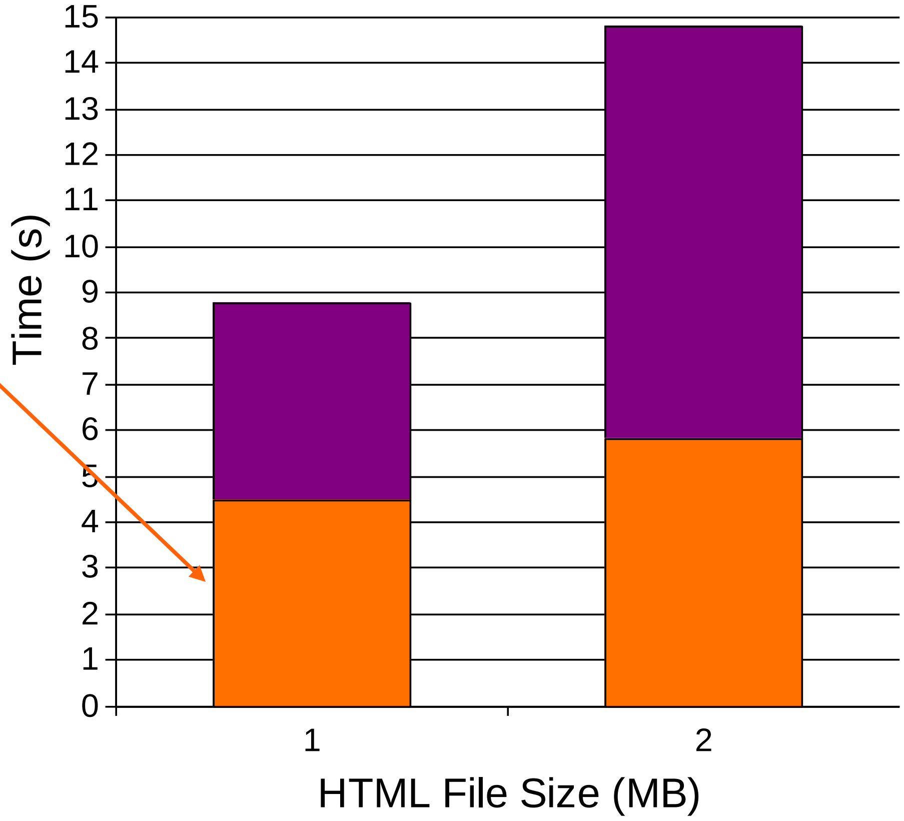
Regular Expressions
Parsing
Web Scripting
Applications

What fraction is HTML compilation?

- 10-40% of time spent in lexing, parsing, syntax-directed translation
- (remainder in layout/rendering)

- loading a fark.com page from disk cache; little JavaScript
- on (old) debug build of FireFox 3

(Informal) Performance of Firefox 3

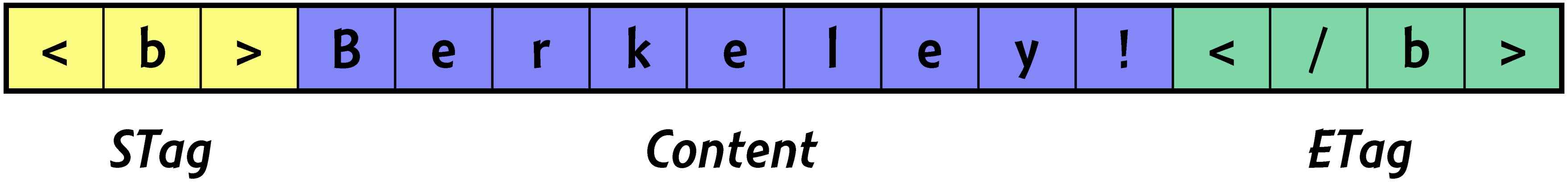


☞ HTML compilation must be parallelized

Lexing, from 10,000 feet

Goal: given lexical specification and input, find lexemes

```
Content ::= [^<]+
ETag    ::= </[^>]*>
STag    ::= <[^>]*>
```



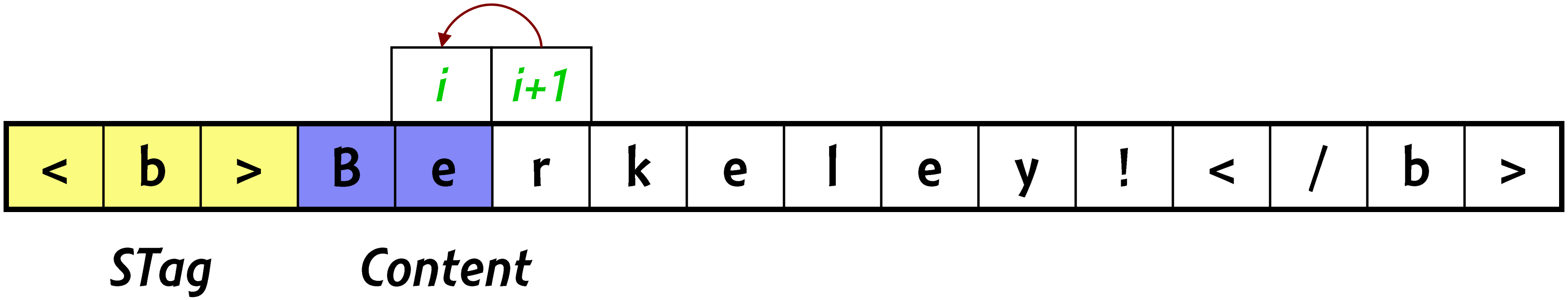
<	b	>	B	e	r	k	e	l	e	y	!	<	/	b	>

STag *Content* *ETag*

Problem: lexing *seems* "inherently sequential"

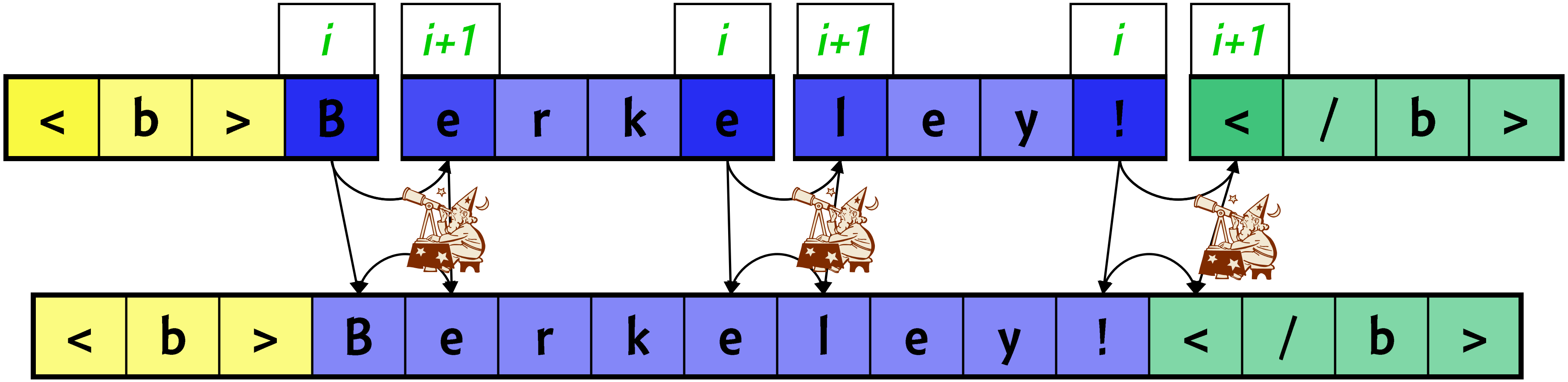
- To know automaton state at input position *i+1*

- We need to know the automaton state at position *i* !



Ideal solution

- Divide input among the processors
- For each processor starting at position *i+1*
  - Ask an **oracle** in which state the neighbor at *i* finished
  - Scan in parallel from next state, at *i+1*
- Finally, merge the results



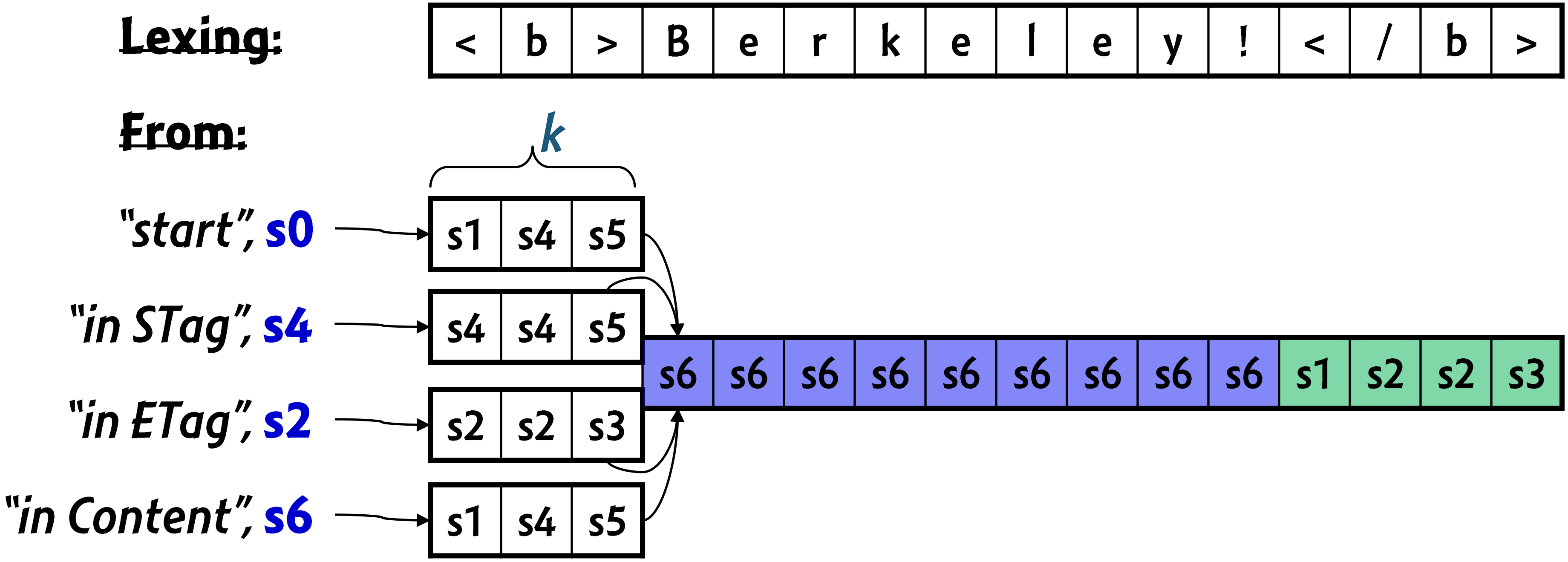
Practical solution: guess! (speculate)

How can we guess from position *$i+1$* the state at position *i*?

- (1) assume state(i) could have been *any* automaton state
  - ☺ the "speculation" is always correct (not really a guess)
  - ☺ can yield O (log *n*) algorithm [Hillis and Steele] …
  - ☹ … but prohibitively expensive in practice
- (2) Was one of a "likely set" of automaton states
  - ☺ can be more efficient than algorithm (1)
  - ☺ can fine-tune speculation based on language and workload
  - ☹ speculation can be wrong
  - ☹ still can be expensive (memory overhead, bad guesses)
- ***But we can do better …***

Our solution (1/2)

Observation: in "real" lexers, the DFA converges to a *stable, recurring* state (often, the "start state"), from multiple initial states, after a small number *k* of characters



Lexing: < b > B e r k e l e y ! < / b >

From:

k

"start", **s0** → s1 s4 s5

"in STag", **s4** → s4 s4 s5

"in ETag", **s2** → s2 s2 s3

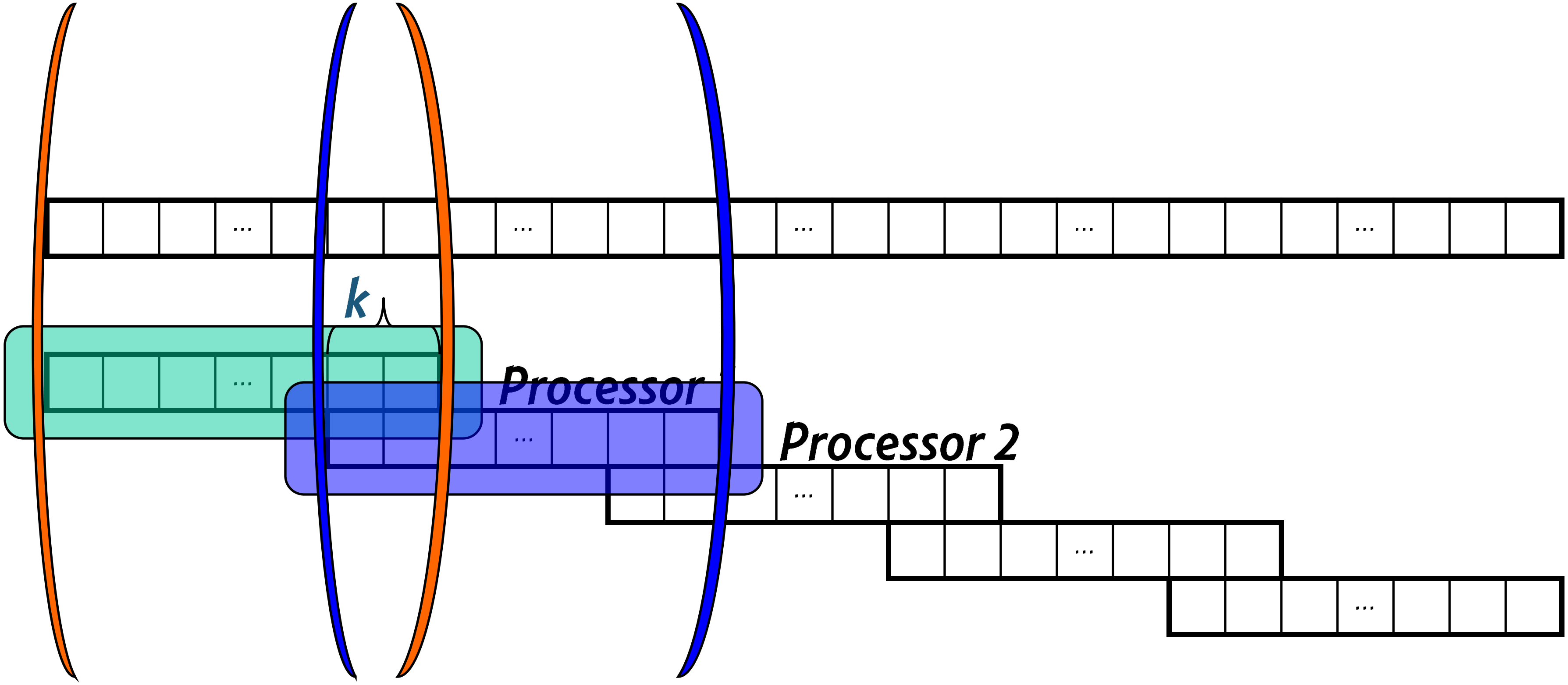
"in Content", **s6** → s1 s4 s5

s6 s6 s6 s6 s6 s6 s6 s6 s6 s1 s2 s2 s3

☞ Only need to follow one DFA path instead of several

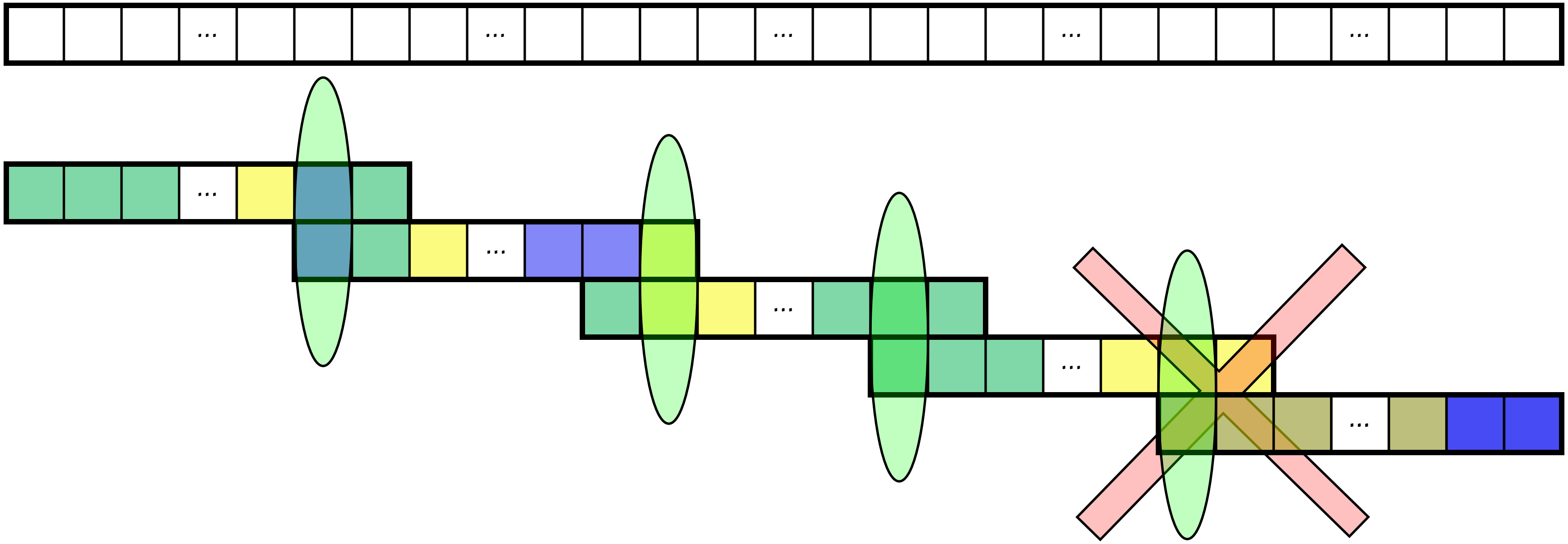
Our solution (2/2)

- **Sketch of our algorithm:**
  - split input into blocks with *k*-character overlap
  - scan blocks in parallel, each starting from “good” initial state



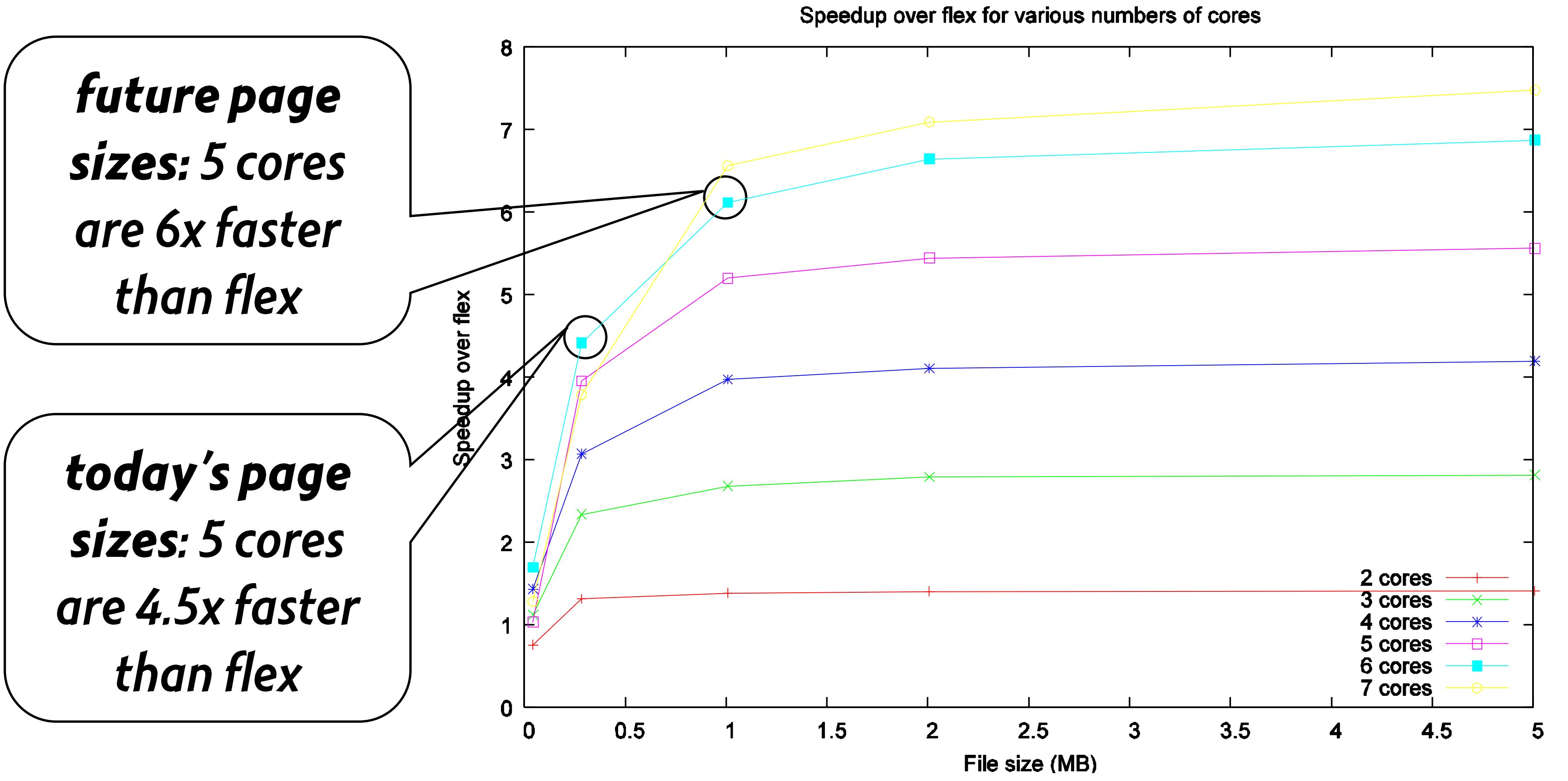
Our solution (2/2)

- **Sketch of our algorithm:**
  - split input into blocks with *k*-character overlap
  - scan blocks in parallel, each starting from "good" initial state
  - find if blocks converge: expected in *k*-overlap
  - speculation may fail; if so, block is rescanned



Preliminary results: speedup over flex

- **flex**: optimized, single-thread lexer on fat Cell core
- Speedup computed by *flex time / cellex time*



***future page sizes:* 5 cores are 6x faster than flex**

***today's page sizes:* 5 cores are 4.5x faster than flex**

Regular expressions

Most teachers, happily cut RE to a single lecture.

We just saw that automata offer fun algorithmic exercises.

Are regular expressions also useful?

- probably the most frequently embedded language
- ex: mashups (extract info from HTML pages, …)

Embedding presents language semantics challenges

- consider the following example …

Example from Jeff Friedl's book

Imagine you want to parse a config file:

```
filesToCompile=a.cpp b.cpp
```

The regex:

```
[a-zA-Z]+=.*
```

Now let's allow an optional \newline-separated 2nd line:

```
filesToCompile=a.cpp b.cpp \[NL] d.cpp e.h
```

We extend the original regex:

```
[a-zA-Z]+=.*(\\\n.*)?
```

This regex does not match our two-line input. Why?

What compiler textbooks don't teach you

First, the *string matching* problem in the Textbook World:

- *"Does a regex r match the **entire** string s?"*
- this is a clean statement and suitable for theoretical study
- here is where equivalence of regex and FSM is defined

The matching problem in the Real World:

- *"Given a string s and a regex r, find a **substring** in s matching r."*

Do you see the language design issue here?

- There may be many such substrings.
- We need to decide **what** substring to find.

It is easy to agree where the substring should start:

- the matched substring should be the **leftmost** match

Two schools of Real World regexes

They differ in where it should end:

Declarative approach: return the longest of all matches

- conceptually, enumerates all matches and returns longest

Operational approach: define behavior of *, | operators

*e** match e maximally such that remainder of regex matches

e|e select leftmost choice while allowing remainder to match

```
filesToCompile=a.cpp b.cpp \[NL] d.cpp e.h

        [a-zA-Z]+=.*(\\\n.*)?
```

These are important differences

- We saw that a non-contrived regex behaves differently
  - personal story: I spent 3 hours debugging a similar regex
  - despite reading the manual carefully
- The operational semantics of *
  - does not guarantee longest match
  - forces the programmer to reason about backtracking
- It seems that backtracking is nice to reason about
  - because it's local: no need to consider the entire regex
  - the cognitive load is actually higher, as it breaks composition

Lessons: WTH did things go wrong?

Likely problem:

- long time ago, someone did not know that NFA can find all matches simultaneously and/or
- NFA can be implemented efficiently

I would like to blame perl,

- but this regex semantics seems older

The theory of finite automata is elegant

- a big success of computer science
- we should make sure that our students know it
- and design clean languages

CYK Parser

Simple context-free-language parser

- running time is O(n^3), space O(n^2)

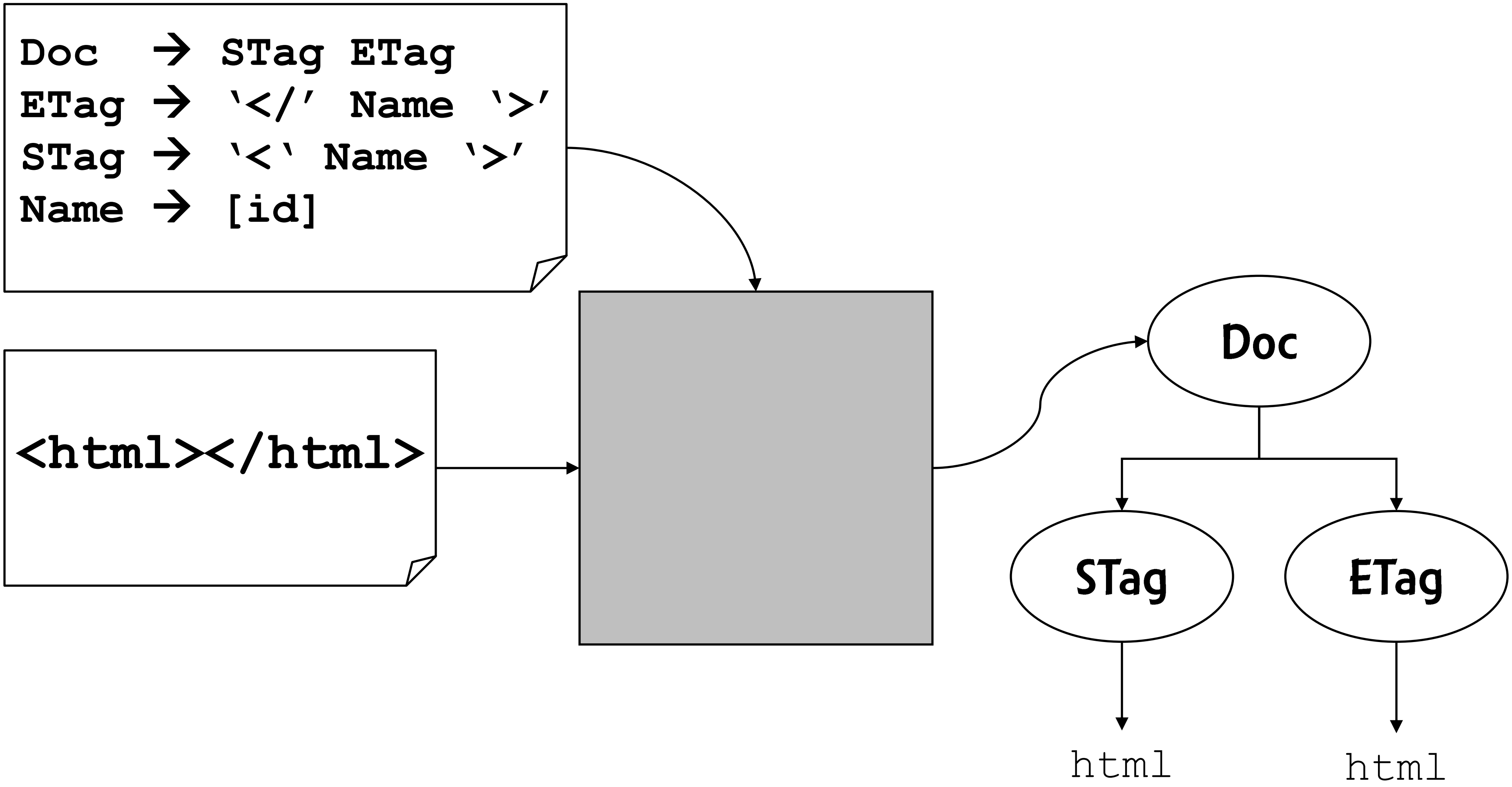
Shunned for many years.

> "Even tabular methods [CYK, Earley] should be avoided if the language at hand has a grammar for which more efficient algorithms [LL, LALR] are available." The Theory of Parsing …, Aho, Ullman, 1972

But in practice, running time is more like $\theta(n^{\approx 1.2})$

- plus computers are now 1,000,000-times faster than in 1972
- browser: we plan to parallelize CYK + Earley parser

Parsing, from 10,000 Feet



```
Doc  → STag ETag
ETag → '</' Name '>'
STag → '<' Name '>'
Name → [id]
```

```
<html></html>
```

Doc

STag

ETag

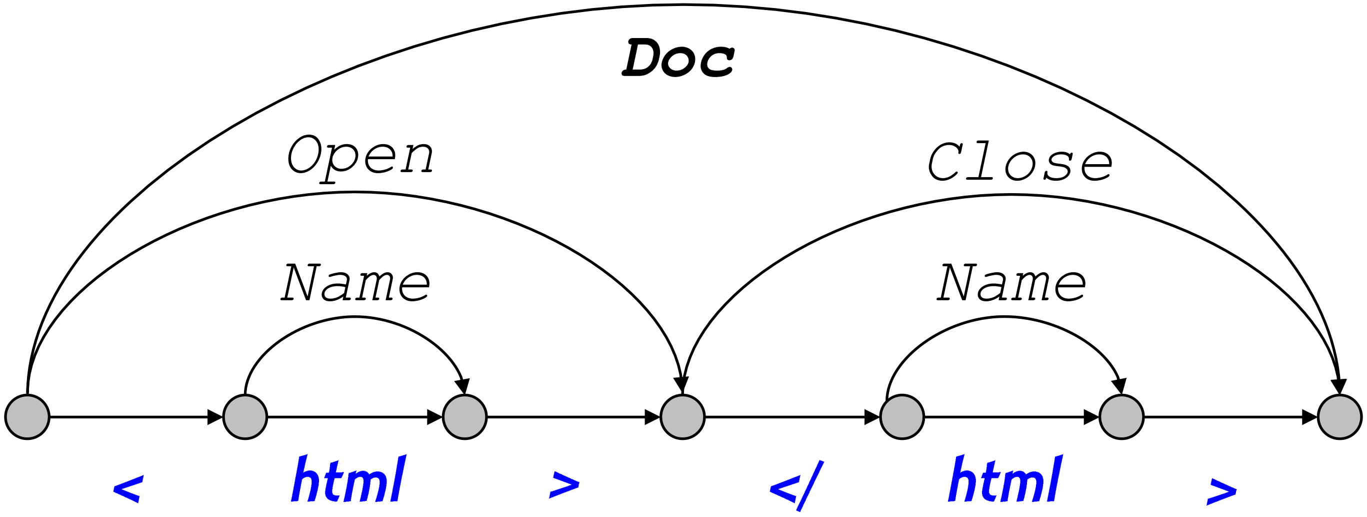
html

html

CYK

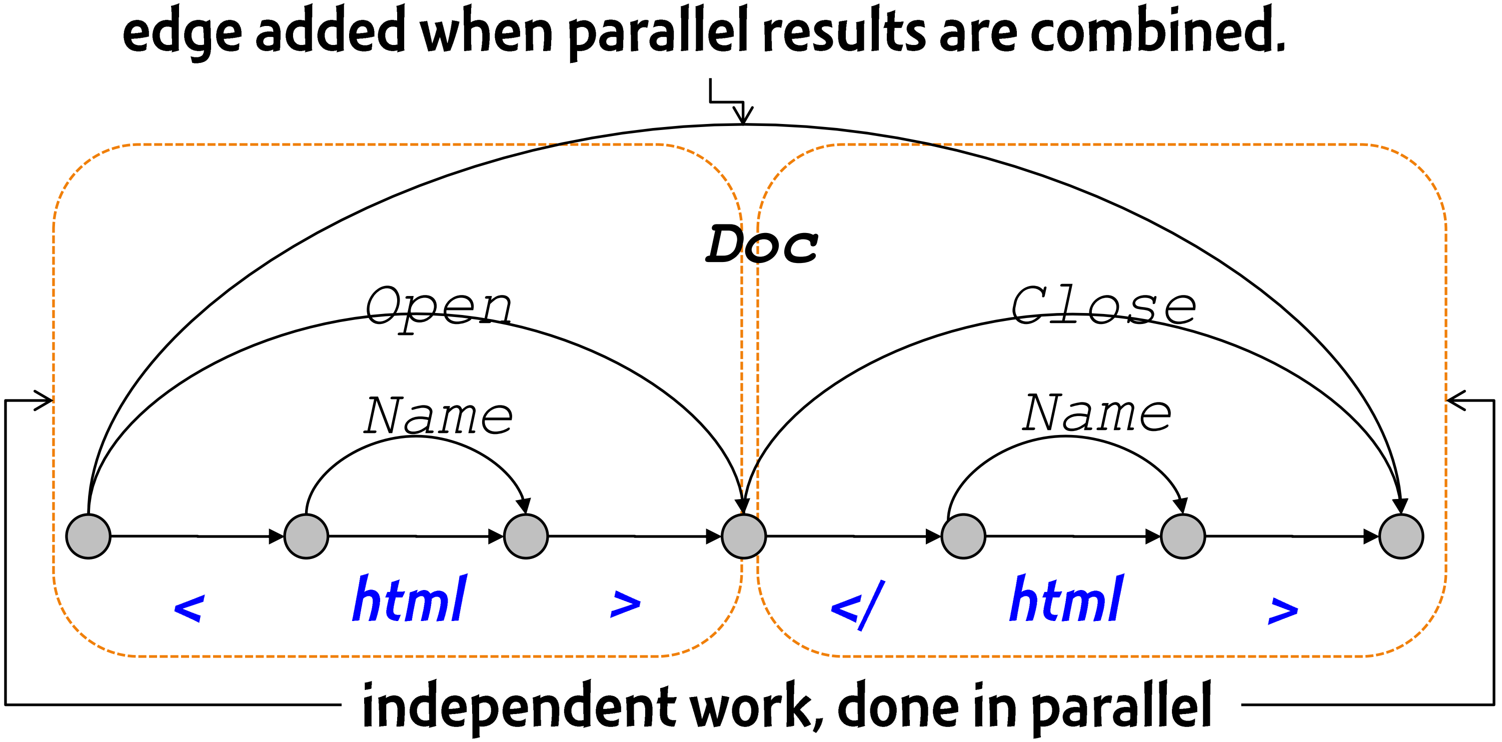
- reduction = adding a non-terminal edge
- reduce until start symbol added

```
Doc   → Open Close
Open  → '<' Name '>'
Close → '</' Name '>'
Name  → [id]
```



Parallel CYK

- CYK is inherently parallel: lots of independent work
  - Bad for serial processing, great for parallel
  - Parse graph is lock-free sparse array ($n \times n \times |G|$)


edge added when parallel results are combined.
Doc
Open
Close
Name
Name
<
html
>
</
html
>
independent work, done in parallel

Parsing in the course

- Back to basics: from LALR to CYK
  - I don't have to relearn LALR each time ☺
- Students used CYK to build their own bison
  - including declarative disambiguation, accepts all grammars
  - teaches ambiguous grammars, needed for real world tasks
  - google calculator:
    - 34 knots in km/h
    - half a dozen pints * (110 Calories per 12 fl oz) / 25W in days
    - 5 in in in
  - live programming (gcalc, PHP in 20 LOC)
- More on parallelism in parsing
  - rethink attribute grammars

A new web client language

AJAX reactive programming is based on clunky callbacks

- too much "plumbing" in the code
- hard to parallelize

Dataflow is a cleaner abstraction

As an example, consider this "follower" program

- where a box follows mouse, with a delay

AJAX code: callbacks

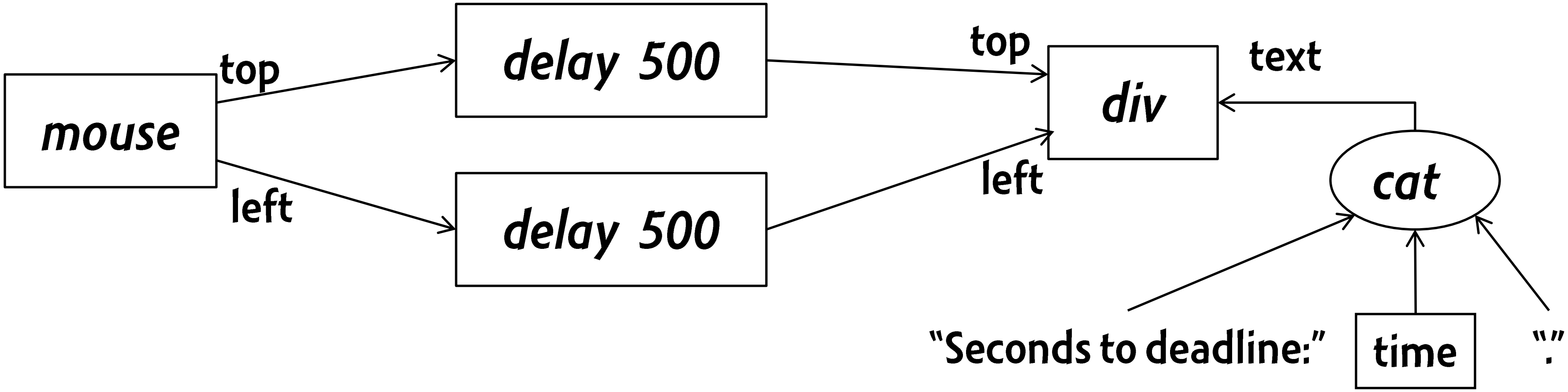
```
<div id="box" style="position:absolute; background: black;">
   Seconds to deadline: <span id="time"> … time … </span>
</div>
<script>

document.addEventListener (
   'mousemove',
   function (e) {
         var left = e.pageX;
         var top = e.pageY;
         setTimeout(function() {
               document.getElementById("box").style.top = top;
               document.getElementById("box").style.left = left;
         } , 500);
   }, false);
 </script>
```

This code moves the box with a delay. We need to set up two nested callbacks. We need to refer to the DOM explicitly by element ids. Code does make it clear at all what's going on. How would you parallelize the program if multiple box were moving on the screen?

FlapJax code: from callbacks to streams

- Program is clearer when data flow in it directly exposed
  - in dataflow version: changing mouse coordinates are streams
  - coordinate streams adjust box position after they are delayed
  - another stream (time) adjusts text after it is formatted

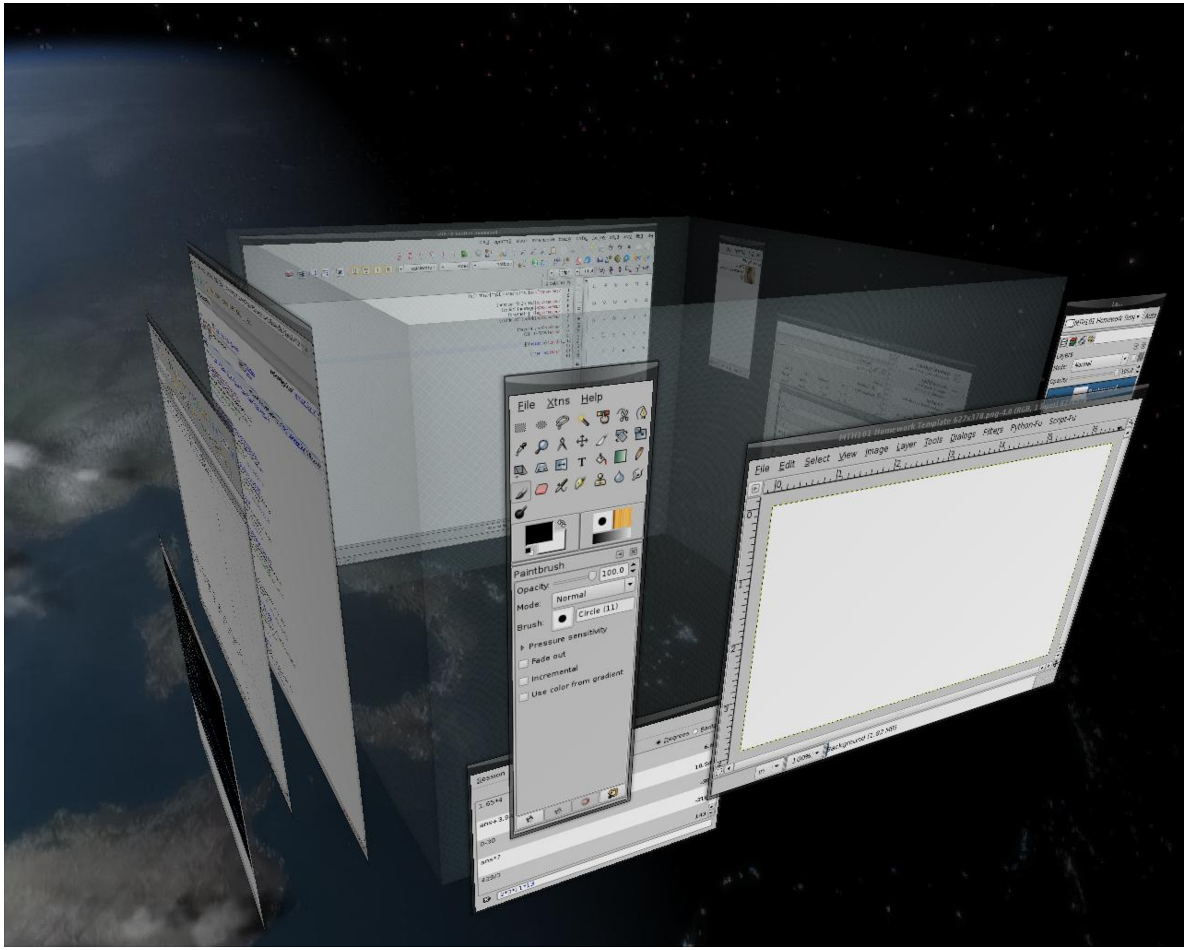


Future apps

- **Future web apps will be like desktop apps and more …**
  - browser = the new windows manager → tabs outdated
  - browser = new OS (local storage, refined security policies)
  - new usage modes (multi-touch, camera-based input, data)
- **We want to identify domains that a browser can support**
  - hypertext documents and media
  - office suites
  - simpler games
  - rich visualization, for data presentation (eg search results)

Example 1: Baryl Desktop Manager

- 3D desktop with physical properties

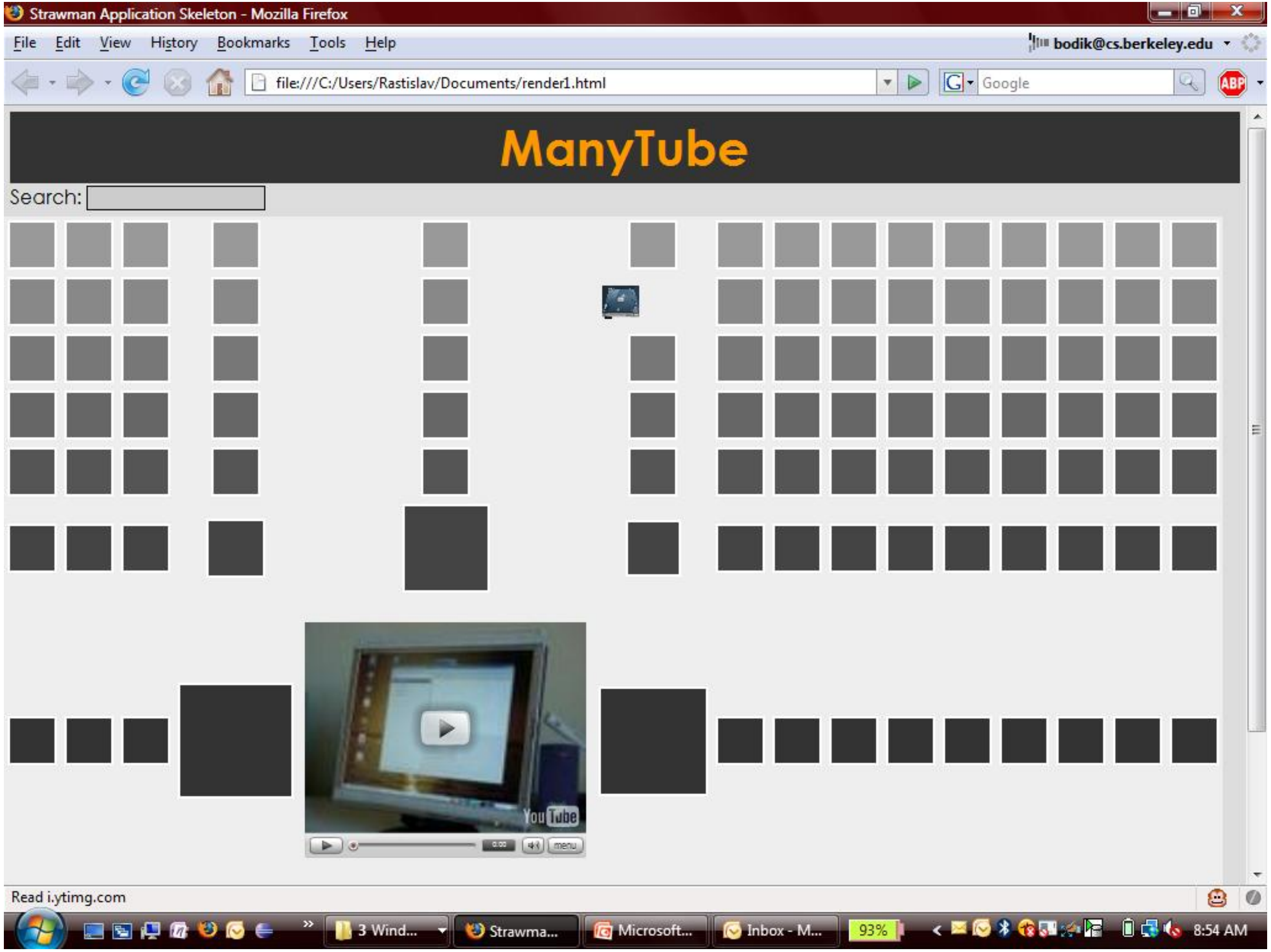


File Xtns Help
File Edit Select View Image Layer Tools Dialogs Filters Python-Fu Script-Fu
Session

Paintbrush
Opacity: 100.0
Mode: Normal
Brush: Circle (11)
▸ Pressure sensitivity
□ Fade out
□ Incremental
□ Use color from gradient

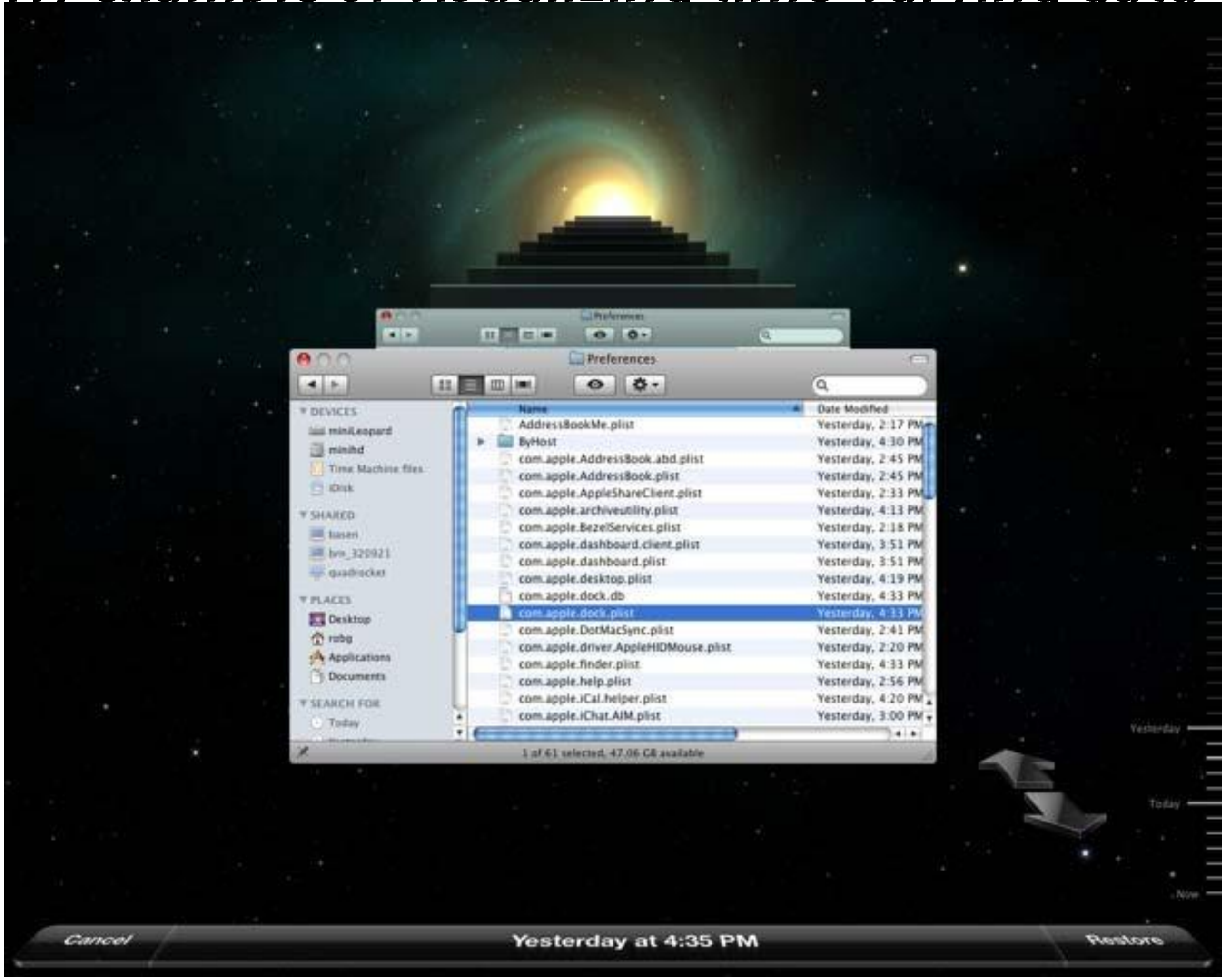
Example 2: ManyTube mockup demo

- Example of a new media app


Strawman Application Skeleton - Mozilla Firefox
File Edit View History Bookmarks Tools Help
bodik@cs.berkeley.edu
file:///C:/Users/Rastislav/Documents/render1.html
Google
ManyTube
Search:
Read i.ytimg.com
3 Wind...
Strawma...
Microsoft...
Inbox - M...
93%
8:54 AM

Example 3: OS X Time Machine

An early example of visualizing time-varying data



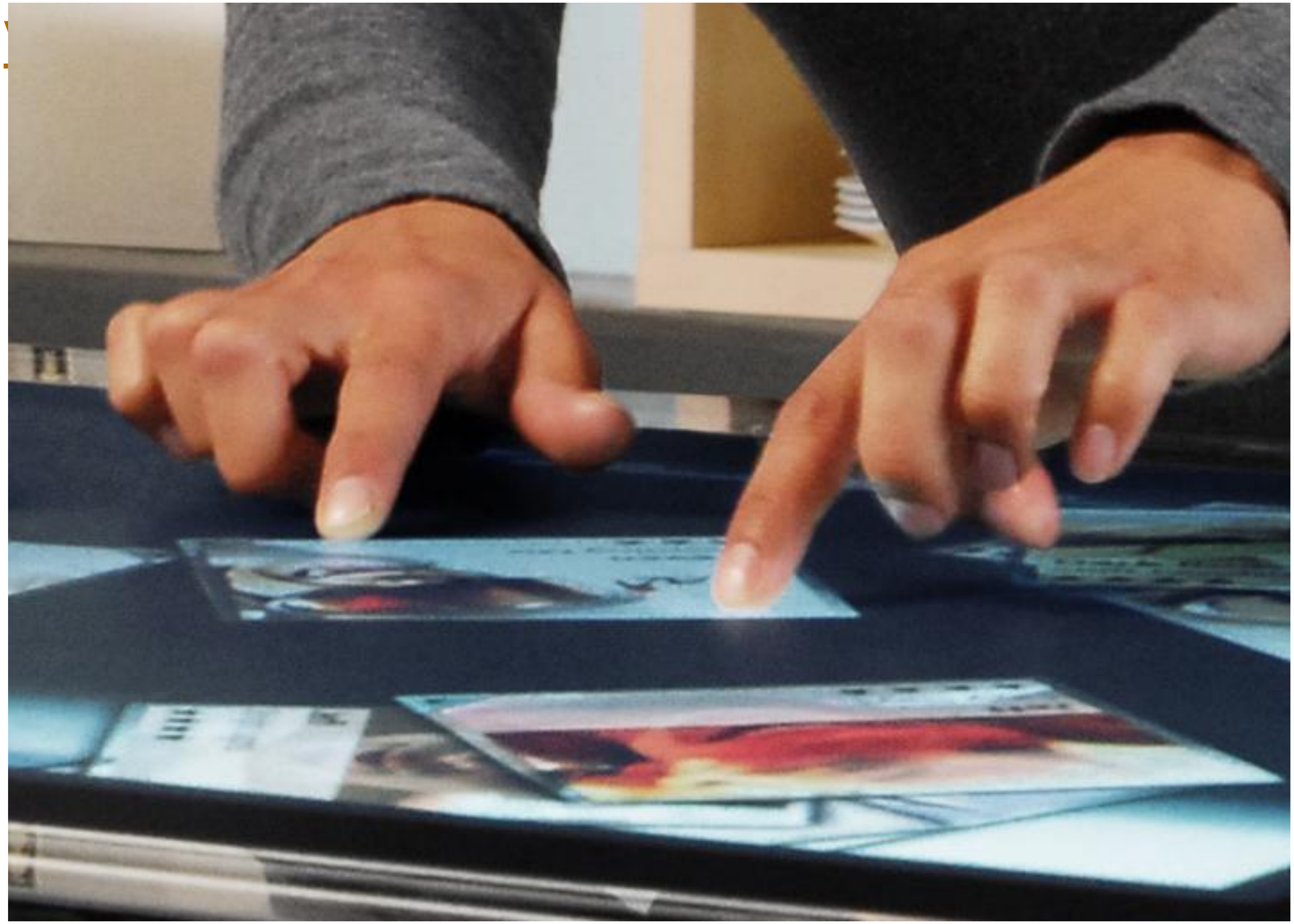
Name	Date Modified
AddressBookMe.plist	Yesterday, 2:17 PM
ByHost	Yesterday, 4:30 PM
com.apple.AddressBook.abd.plist	Yesterday, 2:45 PM
com.apple.AddressBook.plist	Yesterday, 2:45 PM
com.apple.AppleShareClient.plist	Yesterday, 2:33 PM
com.apple.archiveutility.plist	Yesterday, 4:13 PM
com.apple.BezelServices.plist	Yesterday, 2:18 PM
com.apple.dashboard.client.plist	Yesterday, 3:51 PM
com.apple.dashboard.plist	Yesterday, 3:51 PM
com.apple.desktop.plist	Yesterday, 4:19 PM
com.apple.dock.db	Yesterday, 4:33 PM
com.apple.dock.plist	Yesterday, 4:33 PM
com.apple.DotMacSync.plist	Yesterday, 2:41 PM
com.apple.driver.AppleHIDMouse.plist	Yesterday, 2:20 PM
com.apple.finder.plist	Yesterday, 4:33 PM
com.apple.help.plist	Yesterday, 2:56 PM
com.apple.iCal.helper.plist	Yesterday, 4:20 PM
com.apple.iChat.AIM.plist	Yesterday, 3:00 PM

1 of 61 selected, 47.06 GB available

Yesterday at 4:35 PM

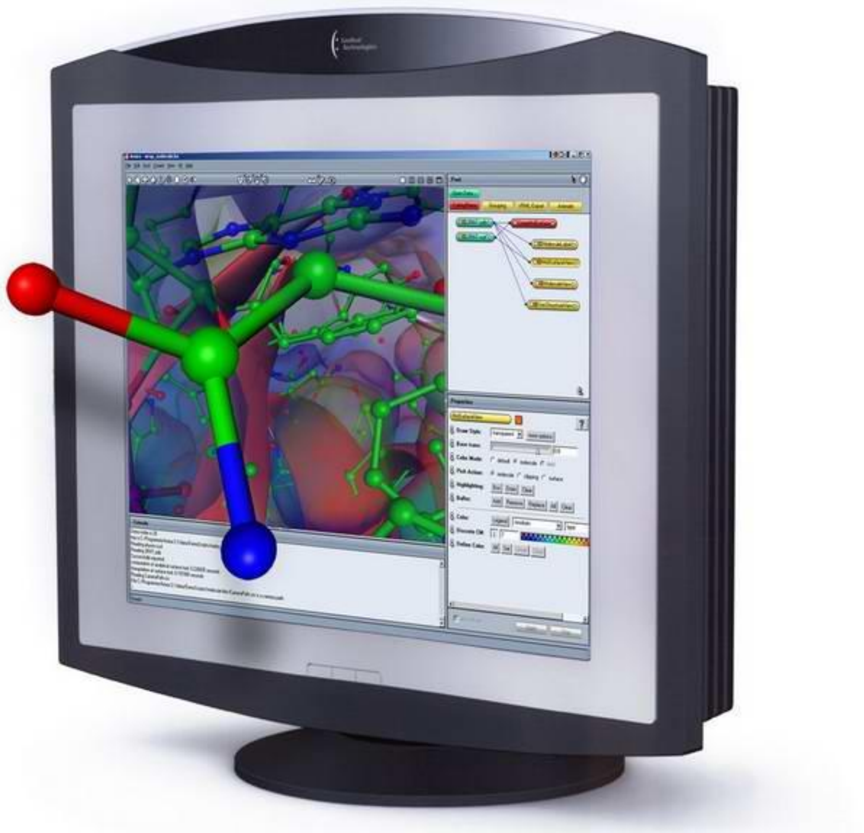
Example 4: Multi-touch interfaces

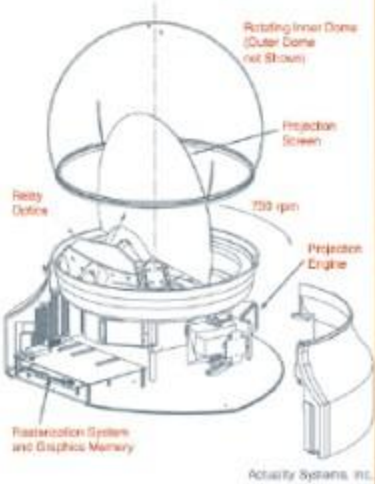
- http://www.youtube.com/watch?v=ysEVYwa-



Example 5: Stereoscopic displays (VR)

- May force us to rethink the desktop metaphor




Rotating Inner Dome (Outer Dome not Shown)
Projection Screen
Relay Optics
730 rpm
Projection Engine
Rasterization System and Graphics Memory
Actuality Systems, Inc.

What should the programming model support?

QoS:

- latency specifications for GUI responsiveness
- video frame rate, etc

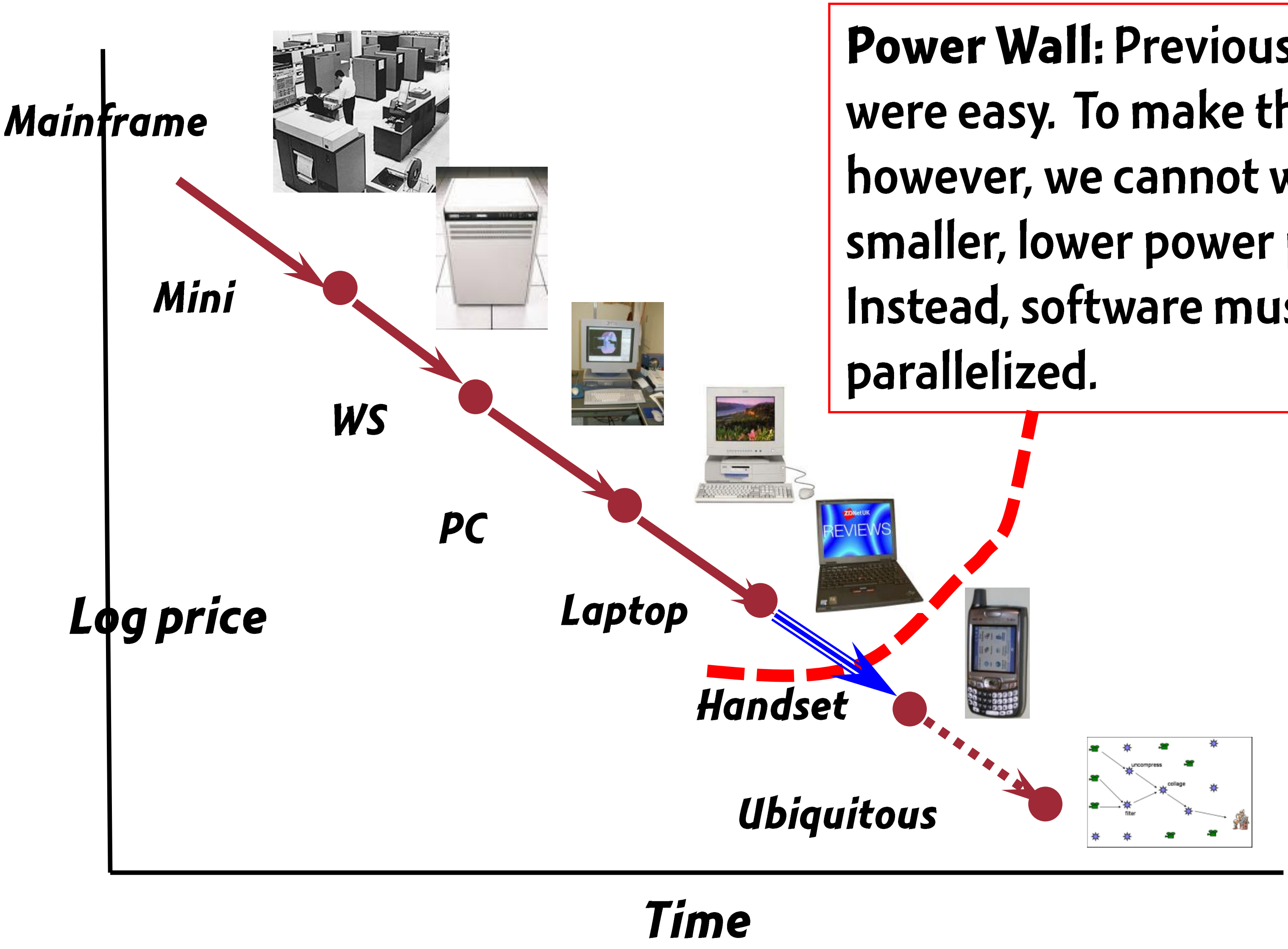
Animation with physical properties (both GUI and games)

- property changes over time, stated **declaratively**
- **trajectories**: how to declare them?
- **physical properties**: stretching, gravity, friction, but maybe also flow, fracture

2.5D and/or 3D

- web page = logical structure + script-produced 3D view?
  - What will a 3D nytimes.com look like? Will 3D ease browsing?
- Q: how to project a part of 3D scene for 2D viewing/reading?

Summary



Mainframe

Mini

WS

PC

Laptop

Handset

Ubiquitous

Log price

Time

Power Wall: Previous Bell steps were easy. To make the net step, however, we cannot wait for smaller, lower power processors. Instead, software must be parallelized.

Conclusion: parallelism

If parallel parsing, layout, and scripting succeeds, browsing on the handheld could be as rich as browsing on the desktop …

… and the next Bell "computer class" will happen.

And all this thanks to advances in langauges and compilation.

Conclusion: undergraduate course

Back to basics

Simpler to teach

Allows students invent new technology,
rather than just learn about it

Need to rethink the course also for parallelism

Backup slides

Course tricks

- debug professor`s code
- submit test cases for grading on a curve
- reinvent CYK
- live programming

Applications

- Drive language development
  - both small in the course
  - and future AJAX in Par Lab
- Course
  - google calculator
  - mashups
  - animation, interaction with flickr
- Future web applications: challenge problems for web client language design