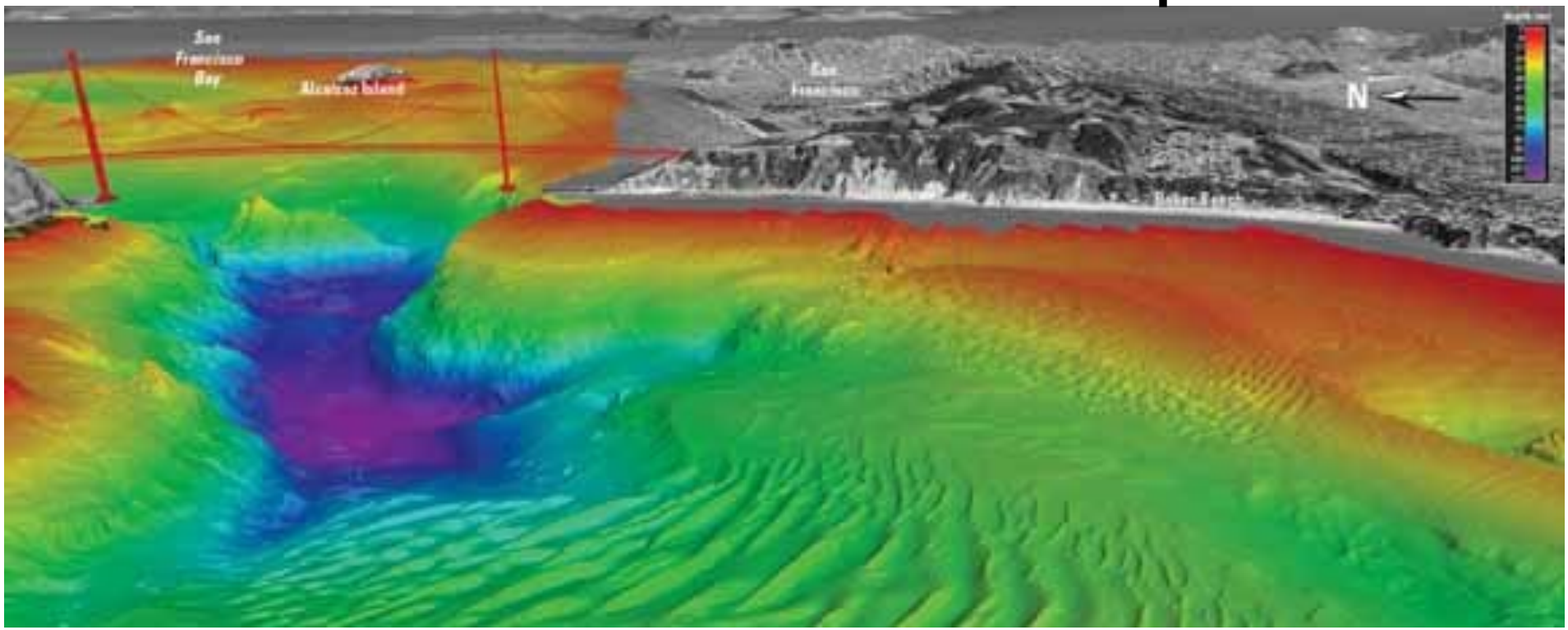
John Rose: JVM Architect, Oracle Corporation

Karl Taylor: J9 GC Team, IBM Canada

The Flattened Data Landscape



From: http://pubs.usgs.gov/sim/2006/2944

"Amateurs talk tactics.
Dilettantes talk strategy.
Professionals talk logistics."

- Military Aphorism

"Bad programmers worry about the code.
Good programmers worry about data
structures and their relationships."

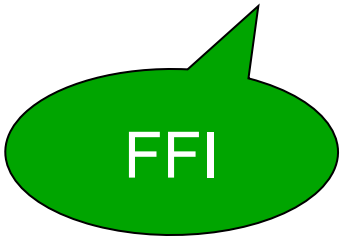
- Linus Torvalds

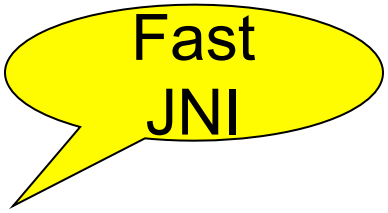
© 2014 IBM Corporation

Important Disclaimers

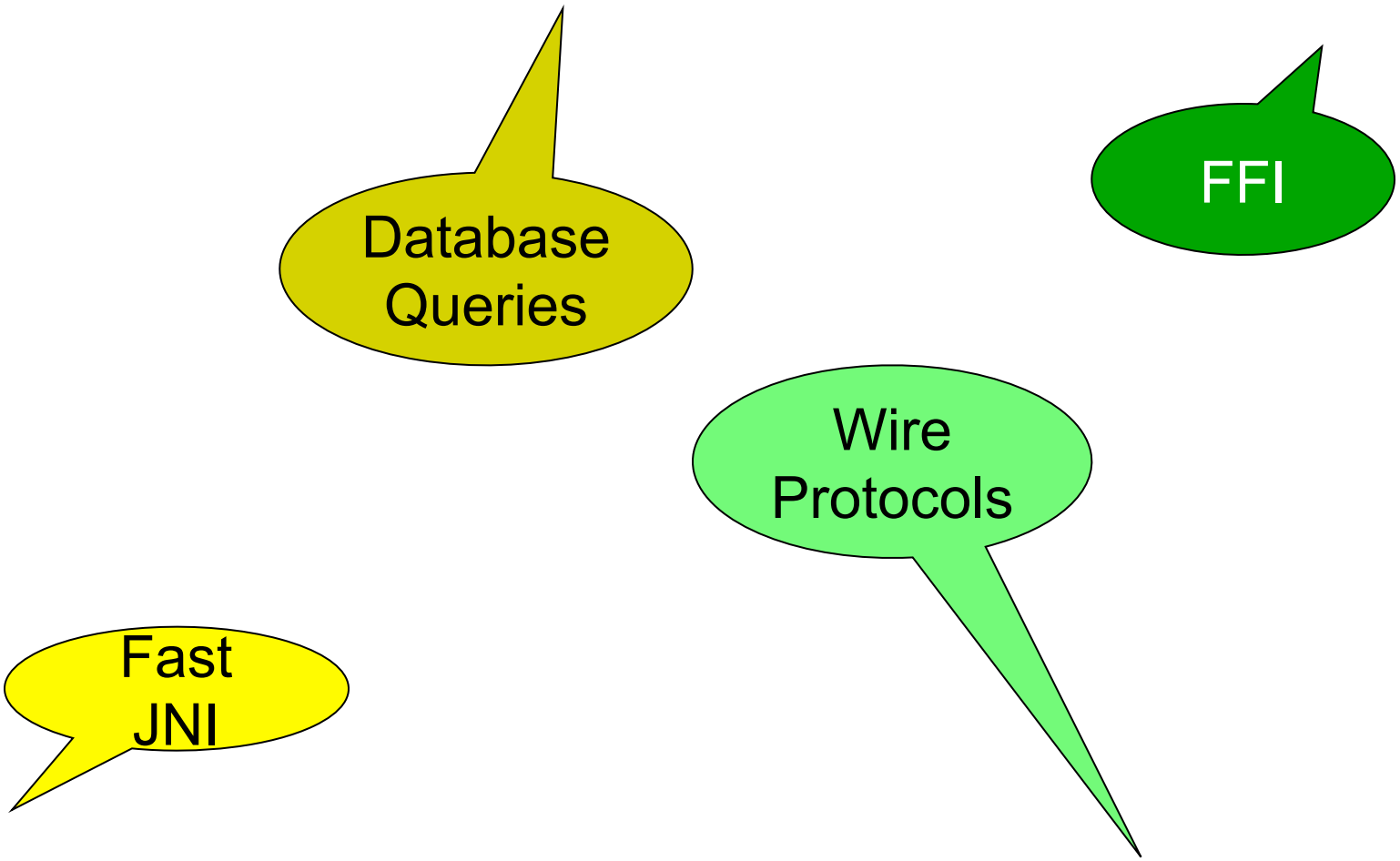
- THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY.
- WHILST EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.
- ALL PERFORMANCE DATA INCLUDED IN THIS PRESENTATION HAVE BEEN GATHERED IN A CONTROLLED ENVIRONMENT. YOUR OWN TEST RESULTS MAY VARY BASED ON HARDWARE, SOFTWARE OR INFRASTRUCTURE DIFFERENCES.
- ALL DATA INCLUDED IN THIS PRESENTATION ARE MEANT TO BE USED ONLY AS A GUIDE.
- IN ADDITION, THE INFORMATION CONTAINED IN THIS PRESENTATION IS BASED ON IBM’S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM, WITHOUT NOTICE.
- IBM AND ITS AFFILIATED COMPANIES SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION.
- NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, OR SHALL HAVE THE EFFECT OF:
- - CREATING ANY WARRANT OR REPRESENTATION FROM IBM, ITS AFFILIATED COMPANIES OR ITS OR THEIR SUPPLIERS AND/OR LICENSORS

© 2014 IBM Corporation

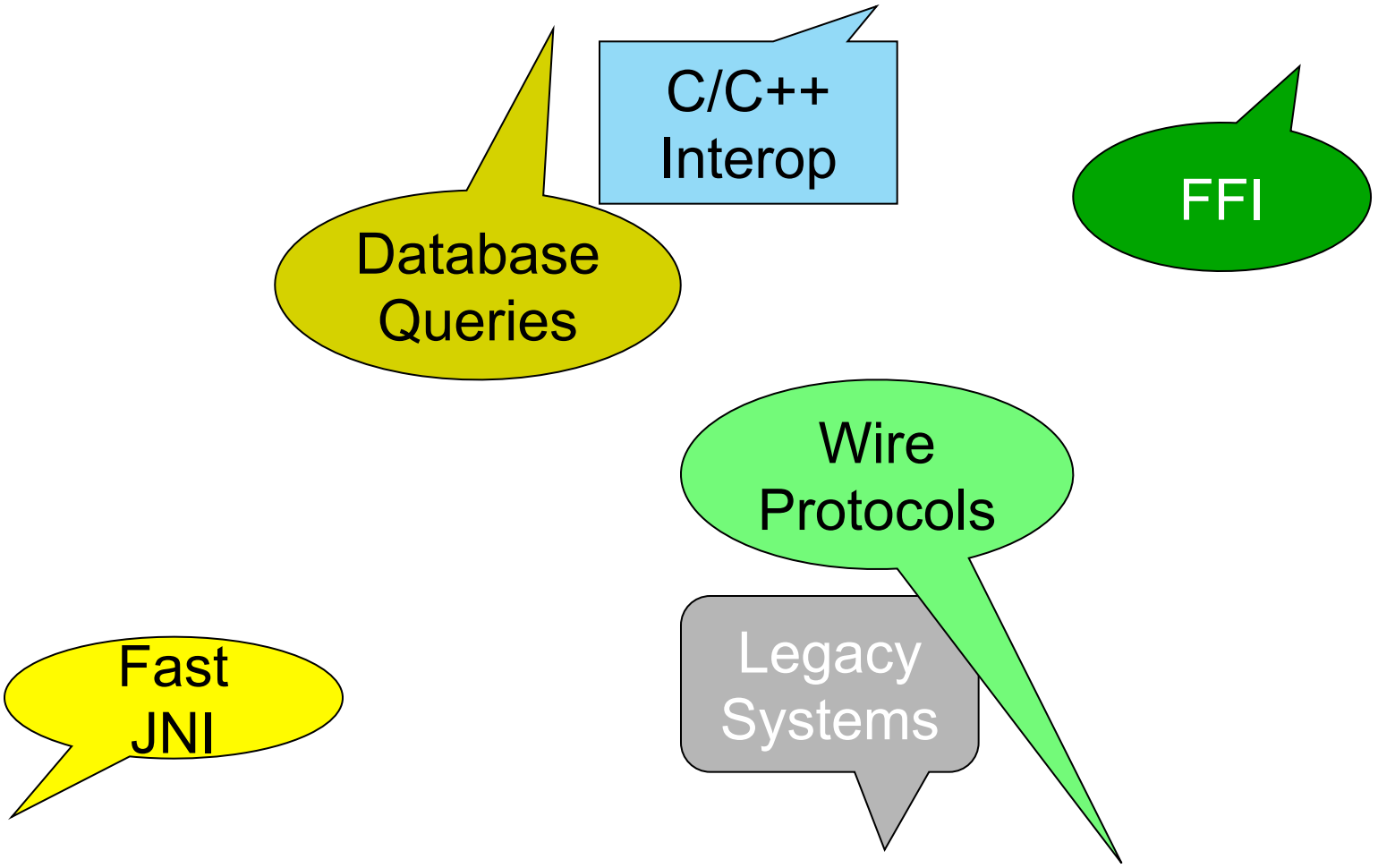




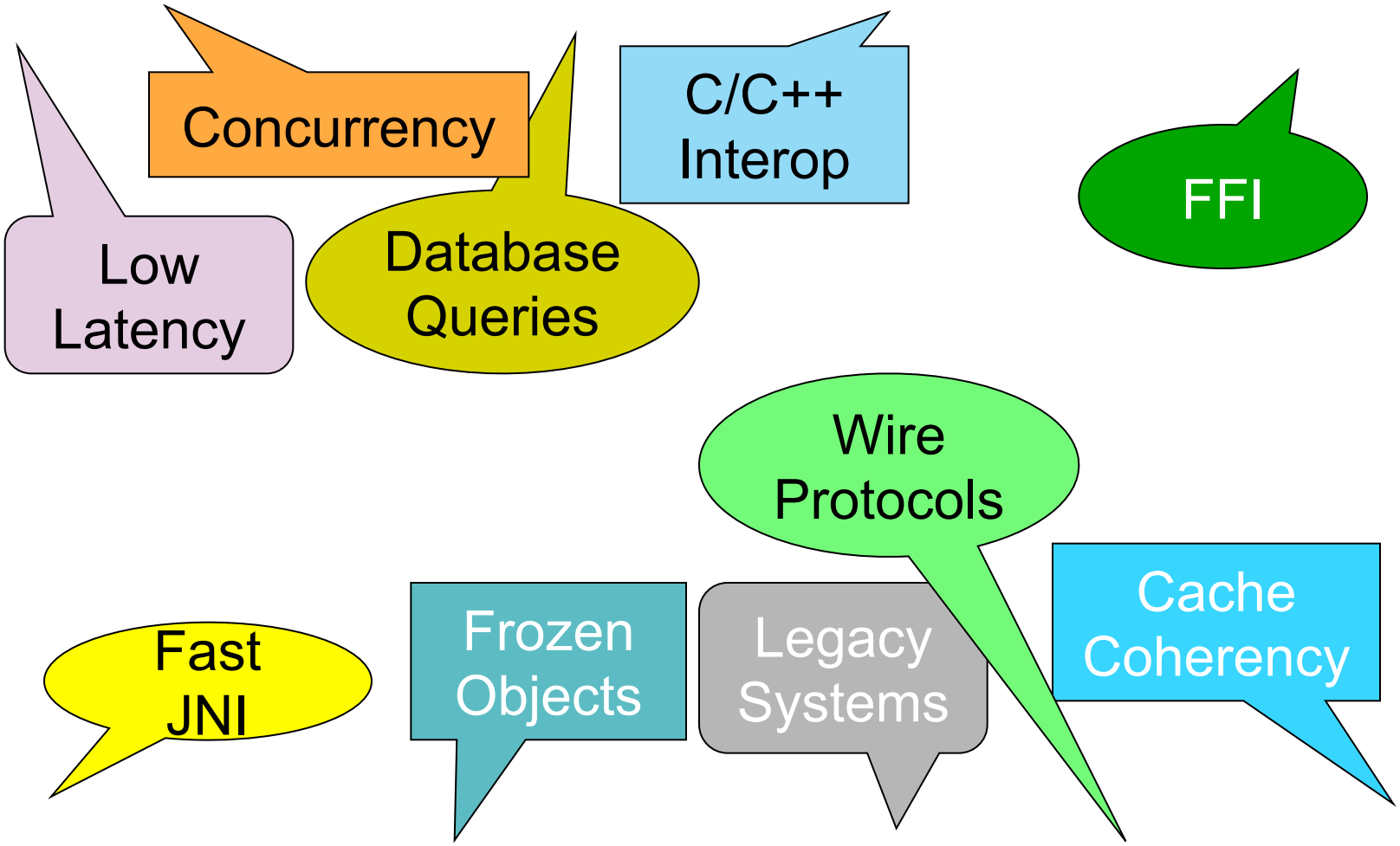
© 2014 IBM Corporation



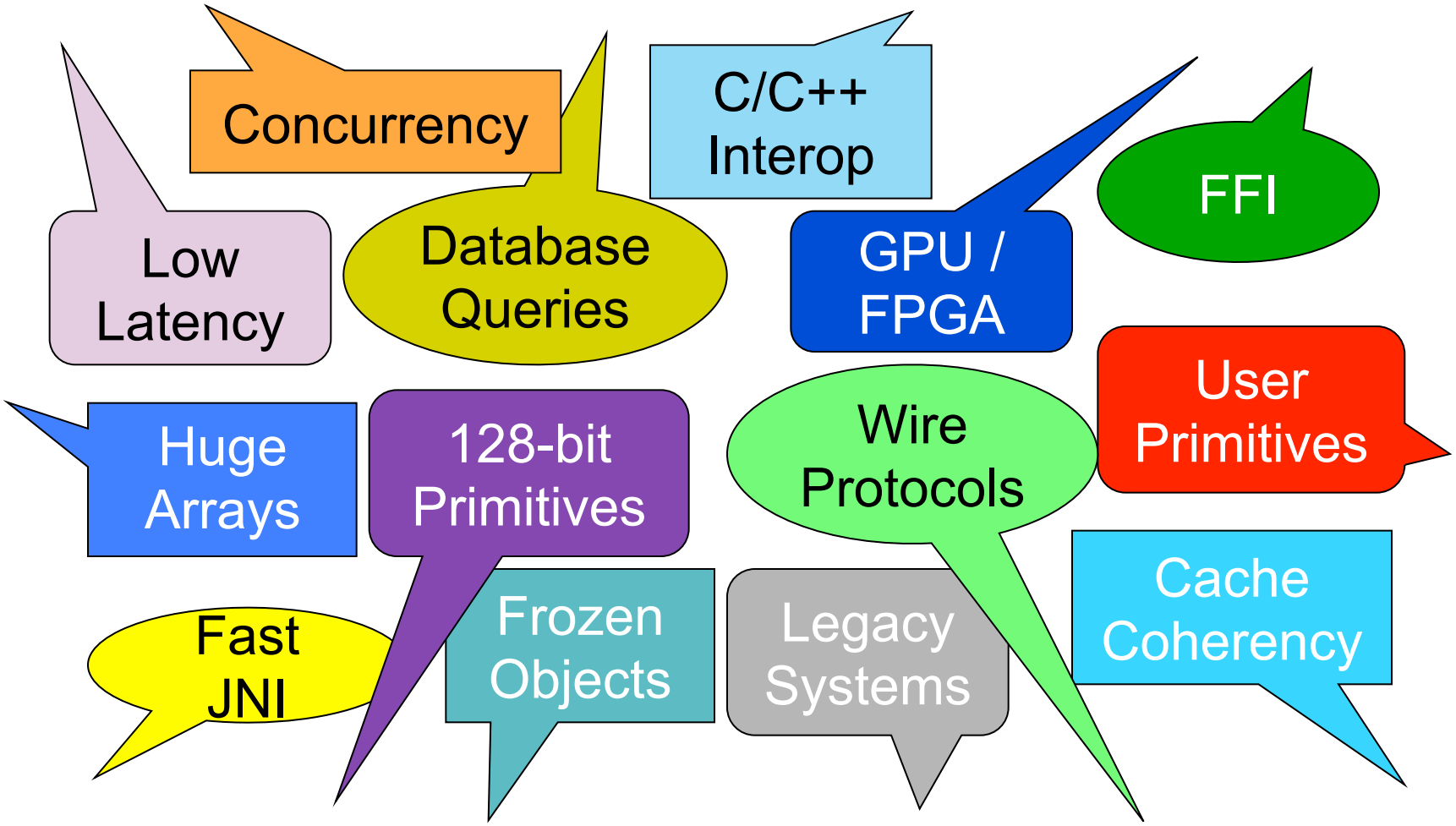
© 2014 IBM Corporation



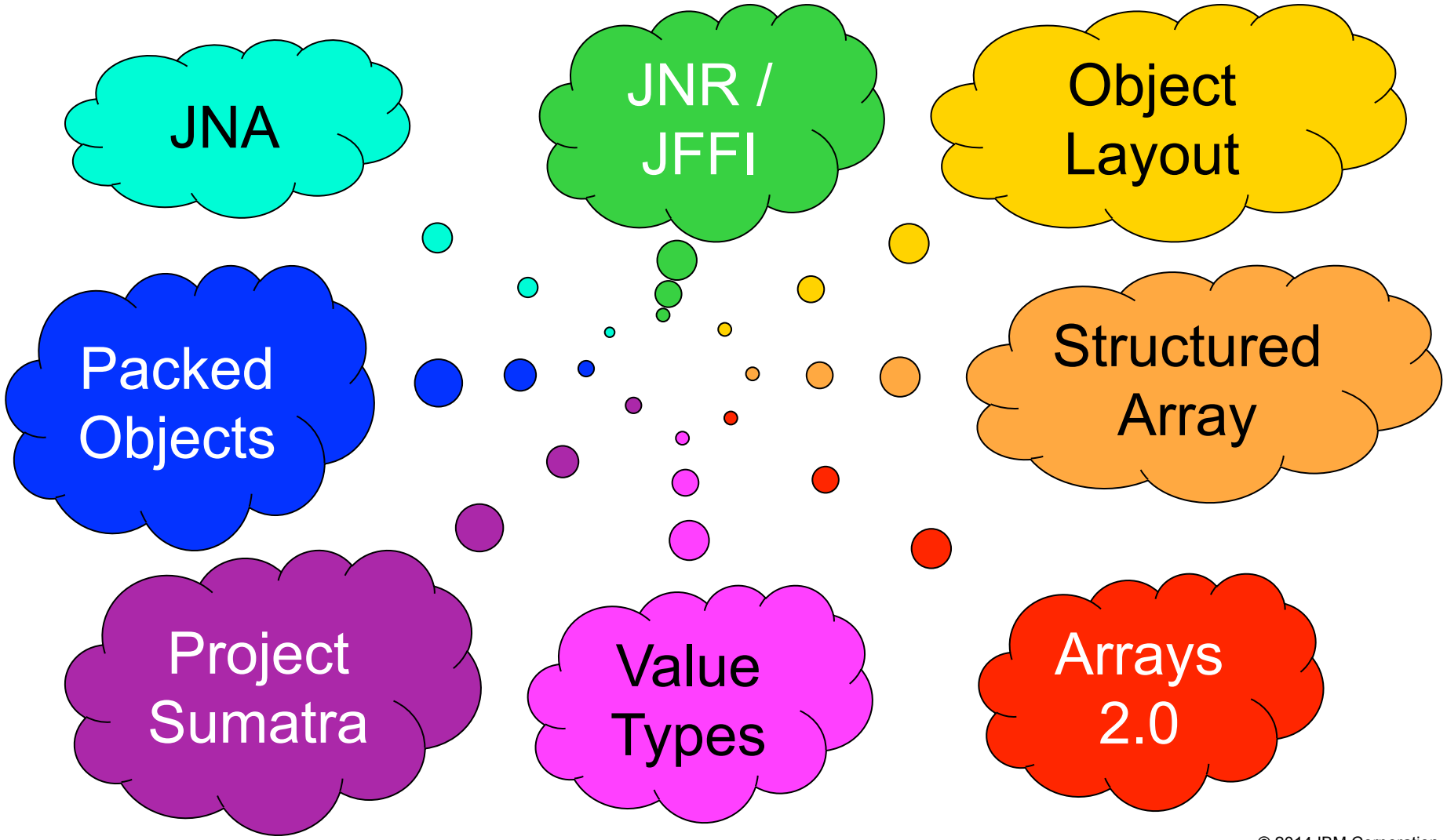
© 2014 IBM Corporation



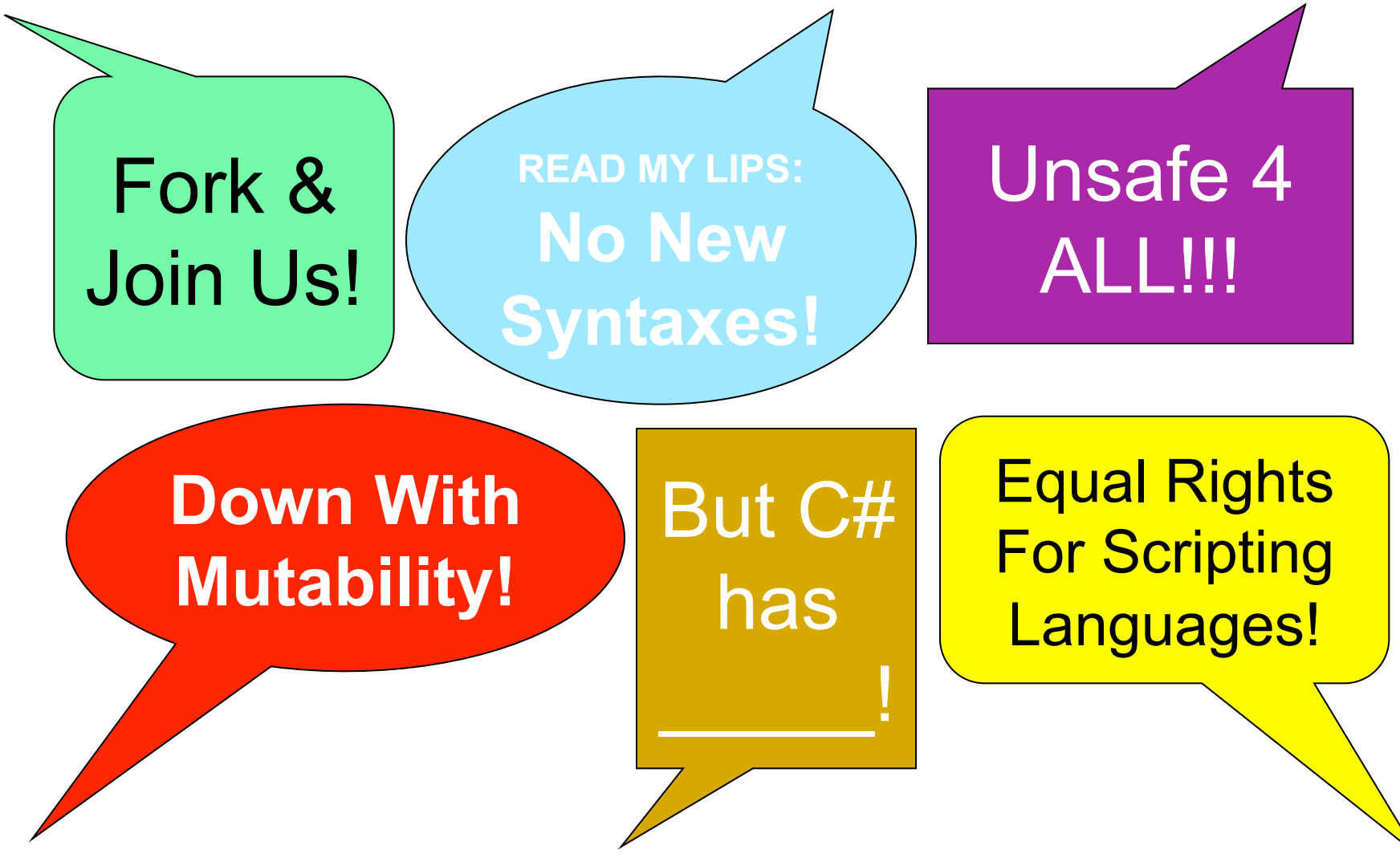
© 2014 IBM Corporation



© 2014 IBM Corporation



© 2014 IBM Corporation



© 2014 IBM Corporation

The Dawn of Data

- Write once read many
- Special tools required
- Excellent security

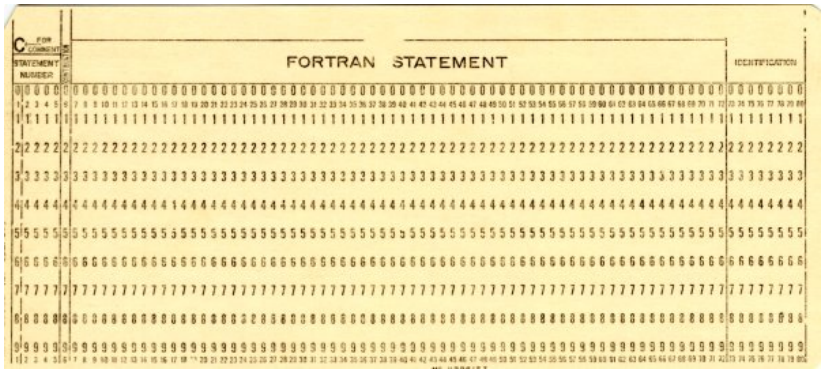
(heavy = hard to steal)



http://commons.wikimedia.org/wiki/File:Rosetta_Stone.JPG
© Hans Hillewaert / CC-BY-SA-3.0

© 2014 IBM Corporation

Early Computing: Cards and tapes



http://commons.wikimedia.org/wiki/File:Hollerith_card.jpg

- Punch cards and magnetic tape
  - Serial access only
  - Used for code and data



http://commons.wikimedia.org/wiki/File:Tapesticker.jpg
CC-BY-SA-3.0

- "Data structure" was a literal statement, not a metaphor

© 2014 IBM Corporation

FORTRAN: Friendly assembly?

- The first popular “high level” language
- Several primitive data types
  - Direct mapping to the underlying hardware
- No user composite types
- Mutability a non-issue?
  - Input and Output were separate concepts

© 2014 IBM Corporation

COBOL: Let me draw you a PICTURE

- Portable by design
- Variables were a fixed number of cells
  - Numeric or alphanumeric
- First composite data types
- No clear mapping to the underlying HW …
  - … but a perfect relationship to databases

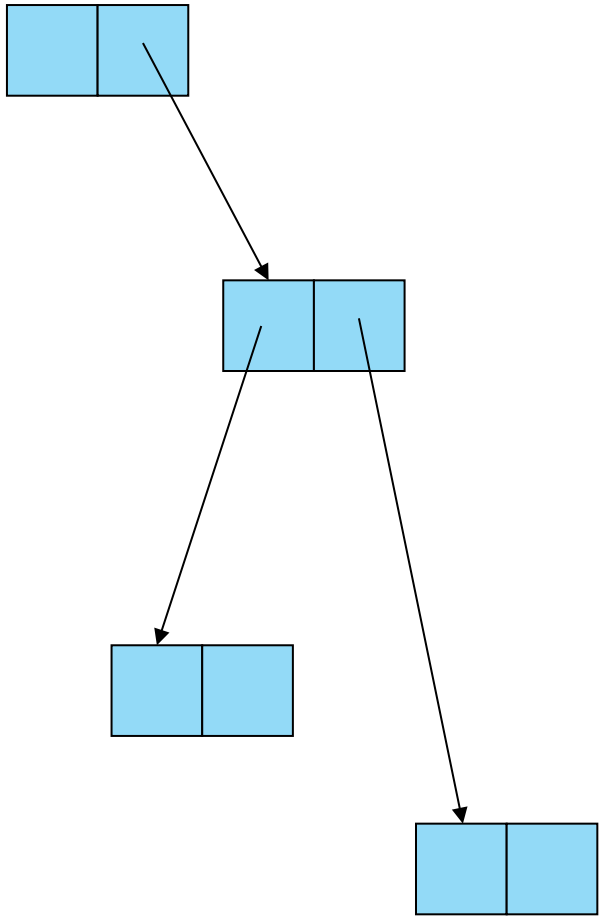
```
01  account.
    03  owner.
        05  lastName    PIC A(30).
        05  firstName   PIC A(30).
        05  uuid        PIC XXXXXXXX.
    03  balance         PIC 9(10)V99.
    03  lastAccessTime  PIC 9(10).
```

© 2014 IBM Corporation

LISP – My other CAR is a CDR…

- Focus on the code, not the data
  - Ease of coding over performance
- No explicit composite types…

… but one infinitely composable type
  - Data structure was an artifact of the code
- Opaque data layout in memory
  - Opening the door for GC



© 2014 IBM Corporation

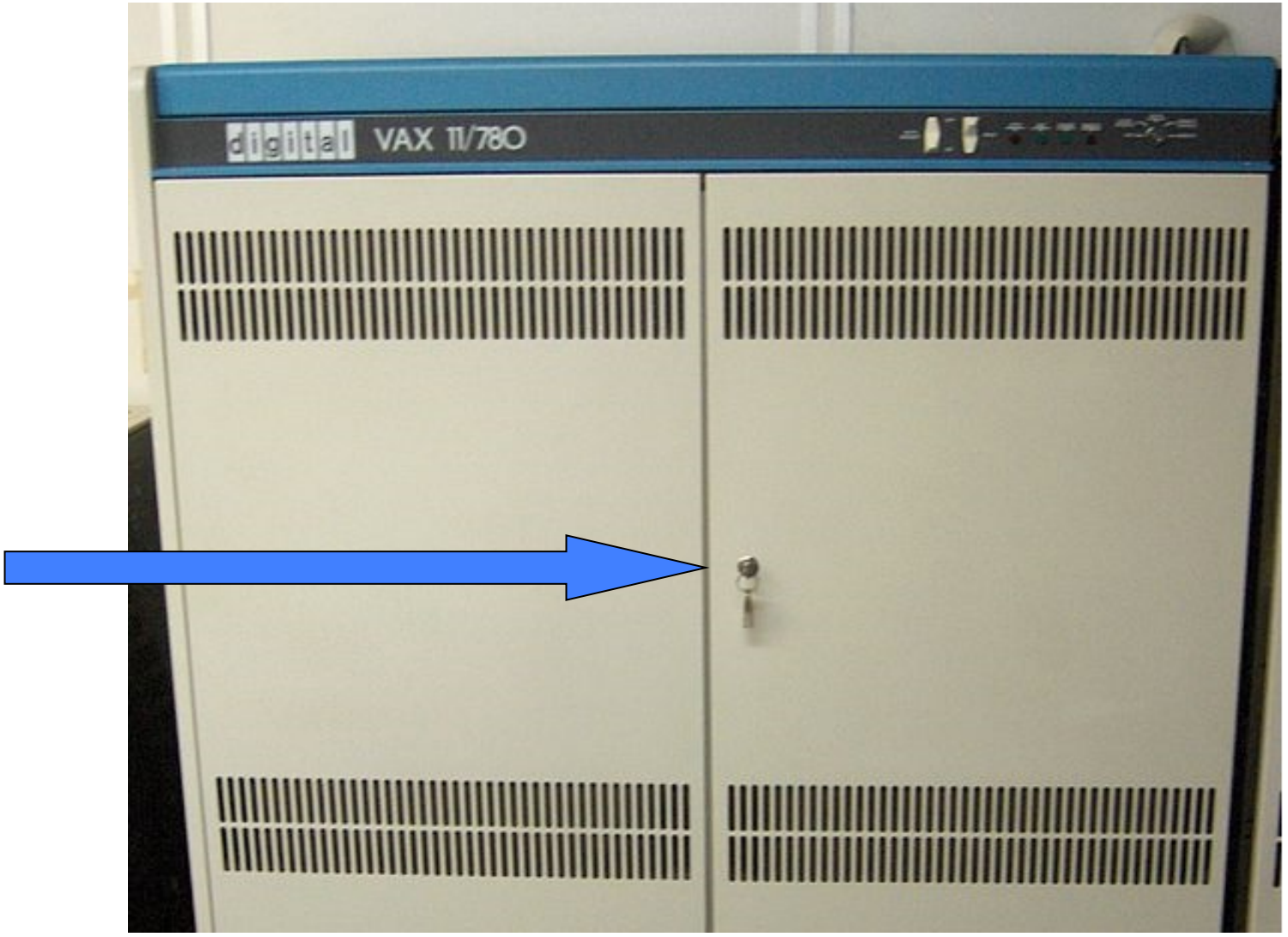
Environment

Many of today's most important concerns

didn't exist in quite the same way…

© 2014 IBM Corporation

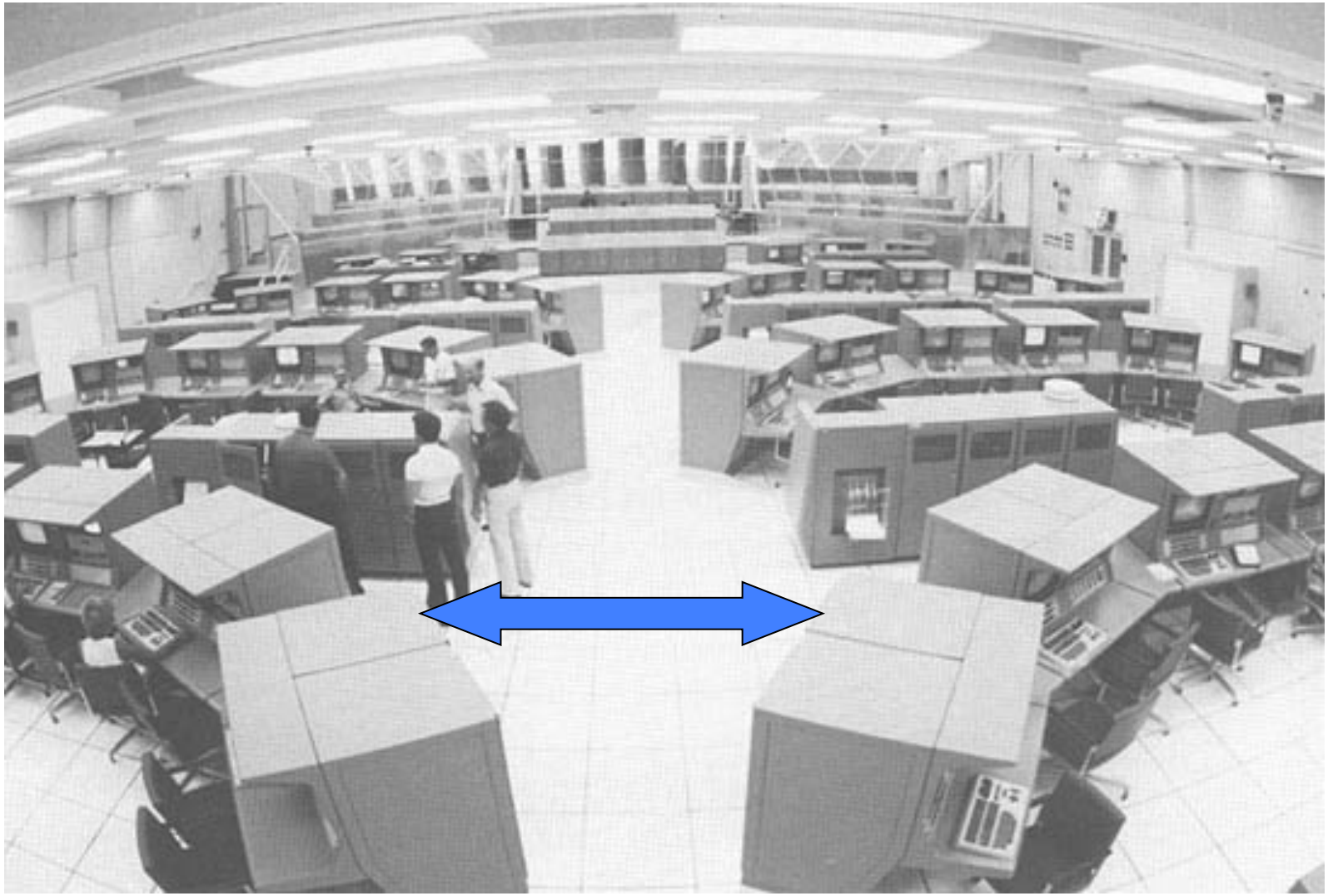
“Security”



http://commons.wikimedia.org/wiki/File:VAX_11-780_intero.jpg

© 2014 IBM Corporation

“Concurrency”



http://history.nasa.gov/computers/Ch7-3.html
NASA photo 108-KSC-78PC-240

© 2014 IBM Corporation

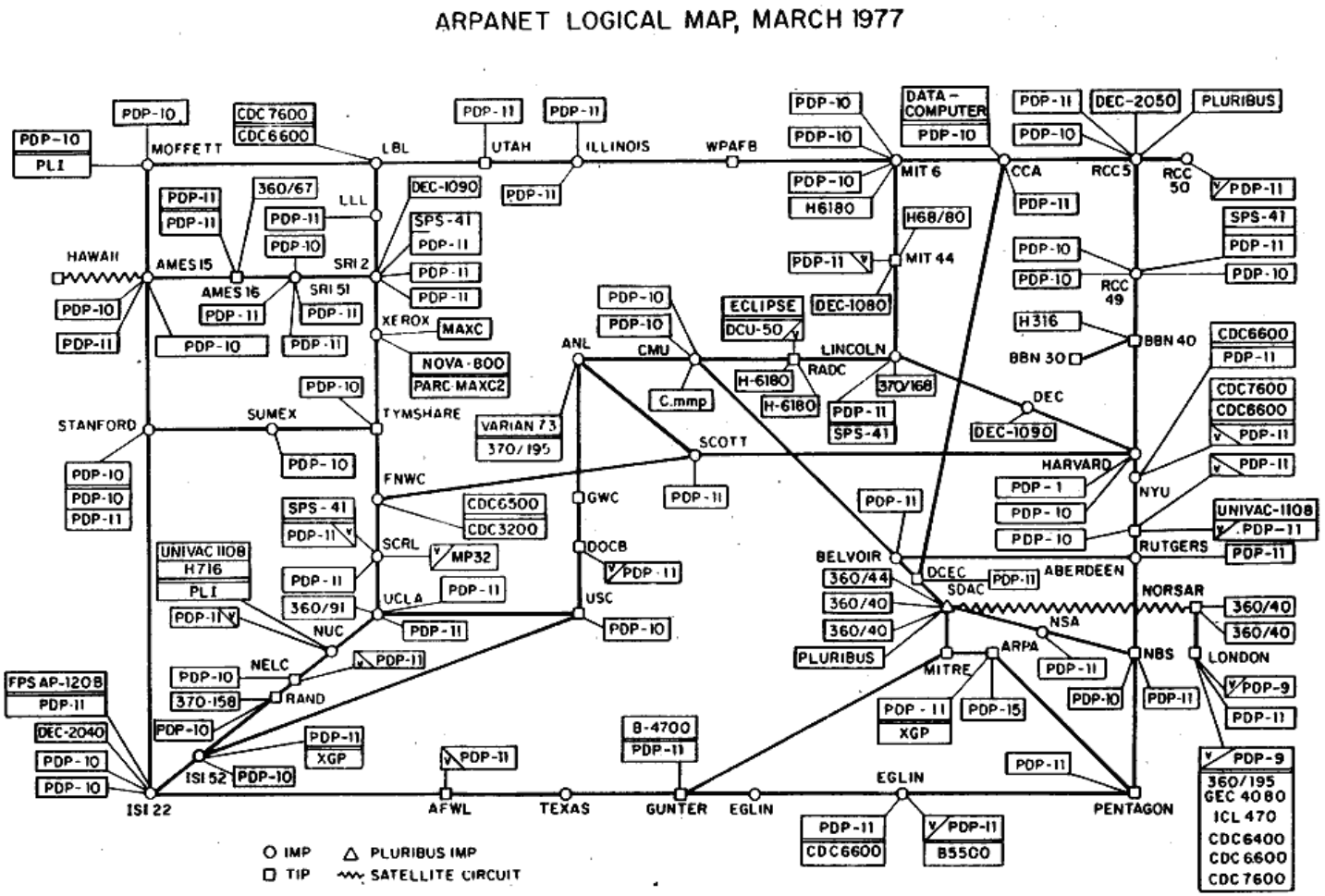
“Mobile Computing”



http://commons.wikimedia.org/wiki/File:IBM_5100_-_MfK_Bern.jpg
CC-BY-SA-3.0

© 2014 IBM Corporation

“Networking”



http://commons.wikimedia.org/wiki/File:Arpanet_logical_map,_march_1977.png

© 2014 IBM Corporation

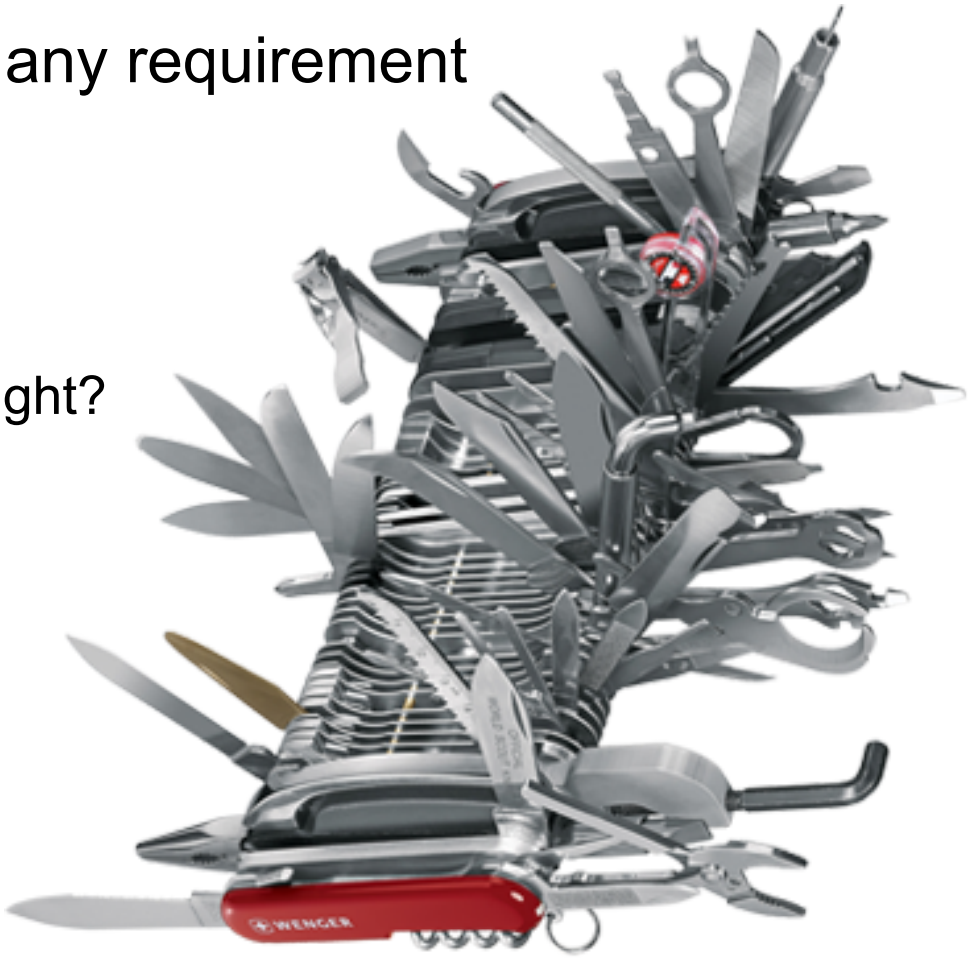
The Big Beast: C

- Inherited FORTRAN's HW-centric types
- Borrowed composite types from COBOL
- Pointer-based structures from LISP
  - But user accessible!
- Portable ***and*** HW-friendly
  - Contradictory, but often just worked

© 2014 IBM Corporation

The Swiss Army Chainsaw

- A tool that can be bent to fit any requirement
  - … and has been!
- Memory is yours to trash
  - You know what you're doing, right?
- Security? Multi-processing?
  - Those are OS problems
- Immutability?
  - Trivially circumvented

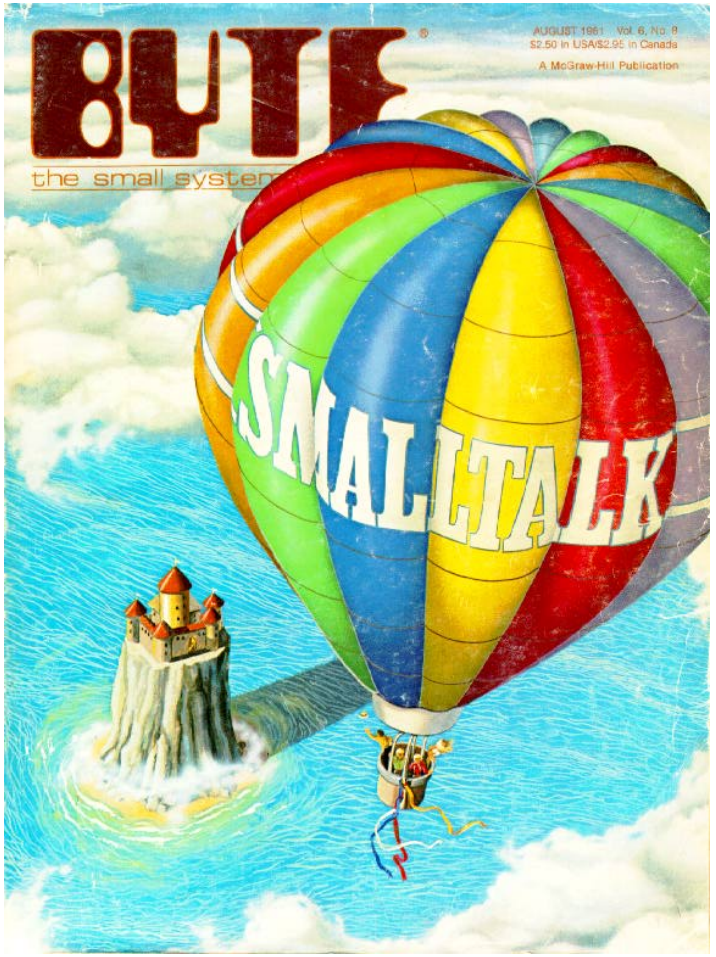


http://www.wengerna.com/giant-knife-16999

© 2014 IBM Corporation

Smalltalk

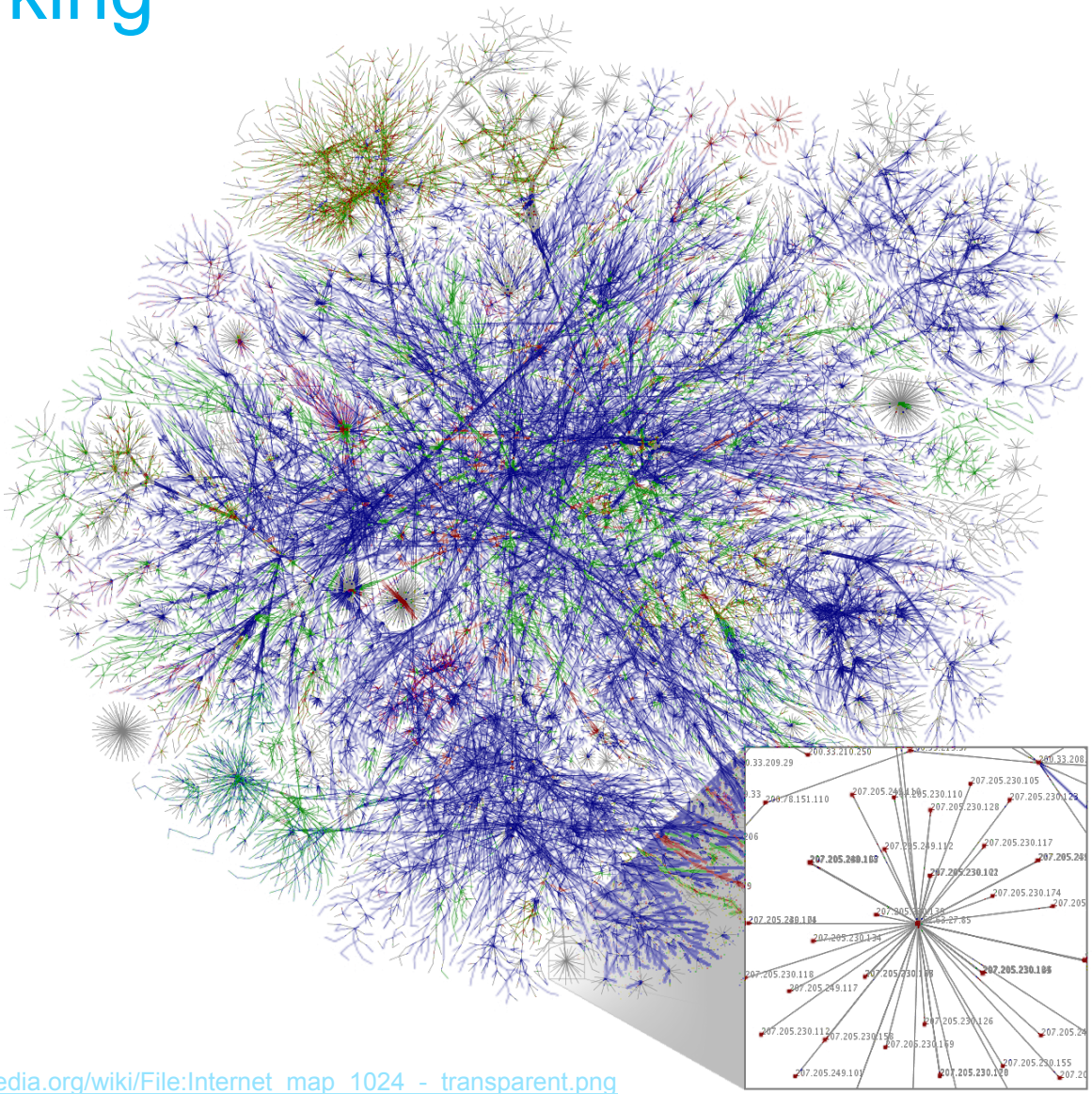
- Another new data paradigm: Objects
- C / COBOL's composite types plus LISP's abstractions
- Stuck in a box
  - … but a nice safe one



http://st-www.cs.illinois.edu/balloon.html
Illustrator: Robert Tinney

© 2014 IBM Corporation

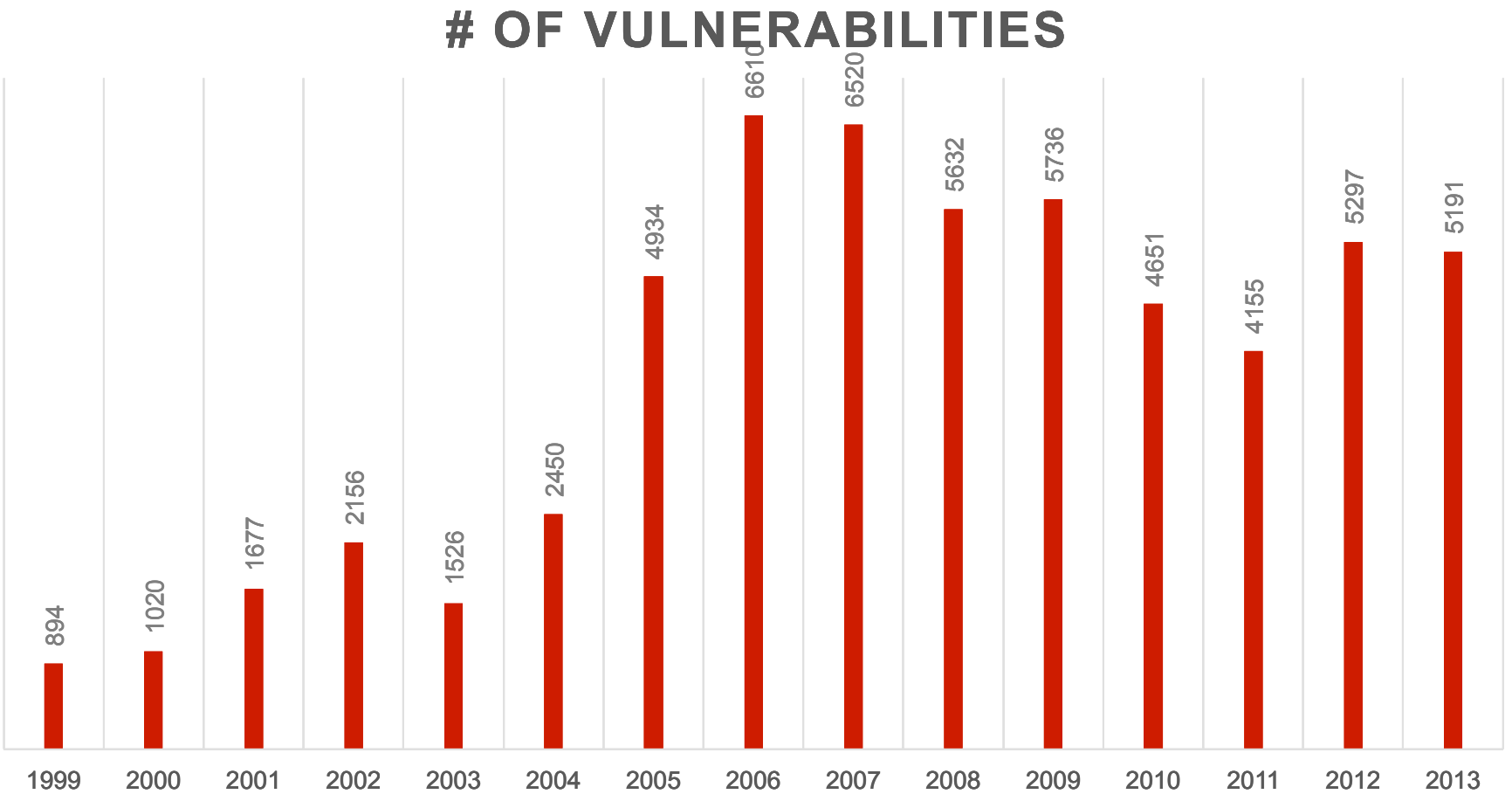
Networking



http://commons.wikimedia.org/wiki/File:Internet_map_1024_-_transparent.png
The Opte Project / CC-BY-2.5

© 2014 IBM Corporation

Security

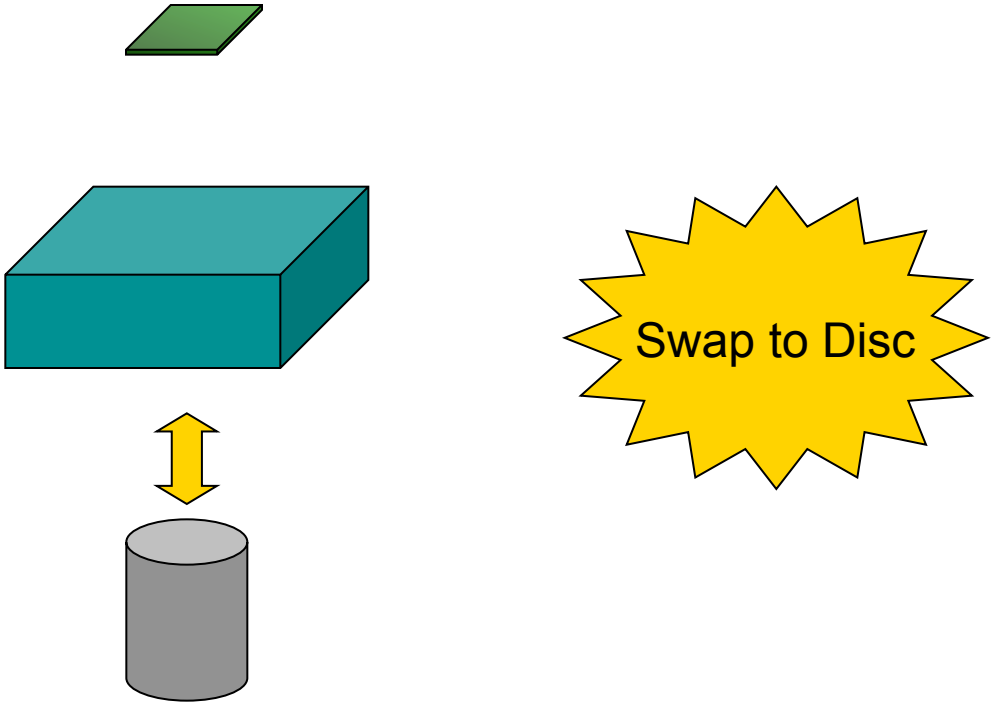


CVE Vulnerabilities By Year

Data from https://cve.mitre.org/

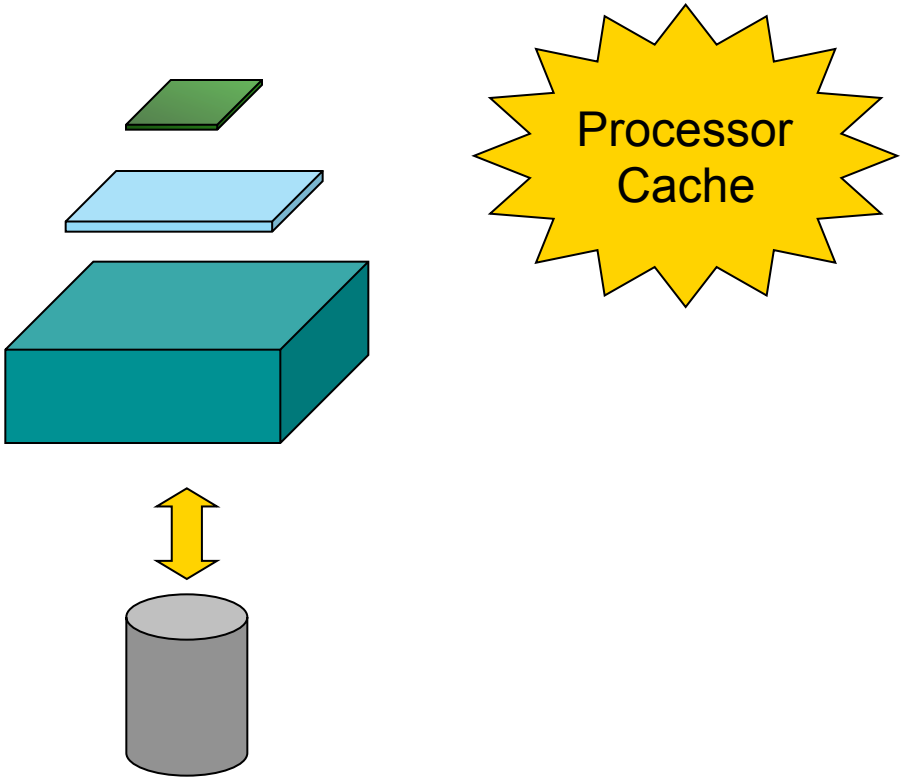
© 2014 IBM Corporation

Hardware Architecture



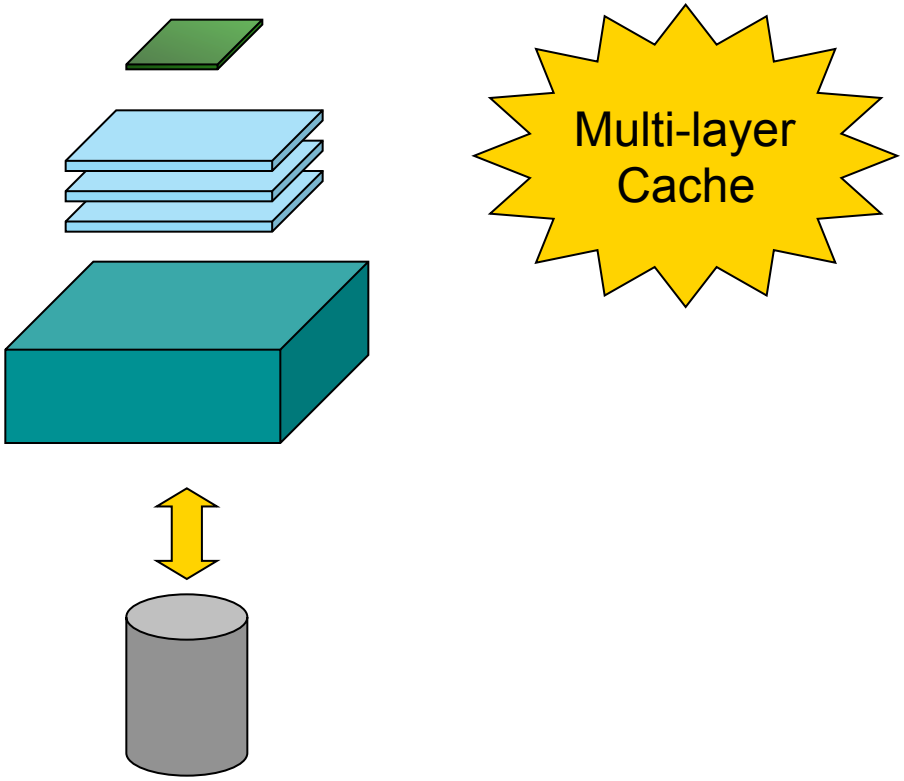
© 2014 IBM Corporation

Hardware Architecture



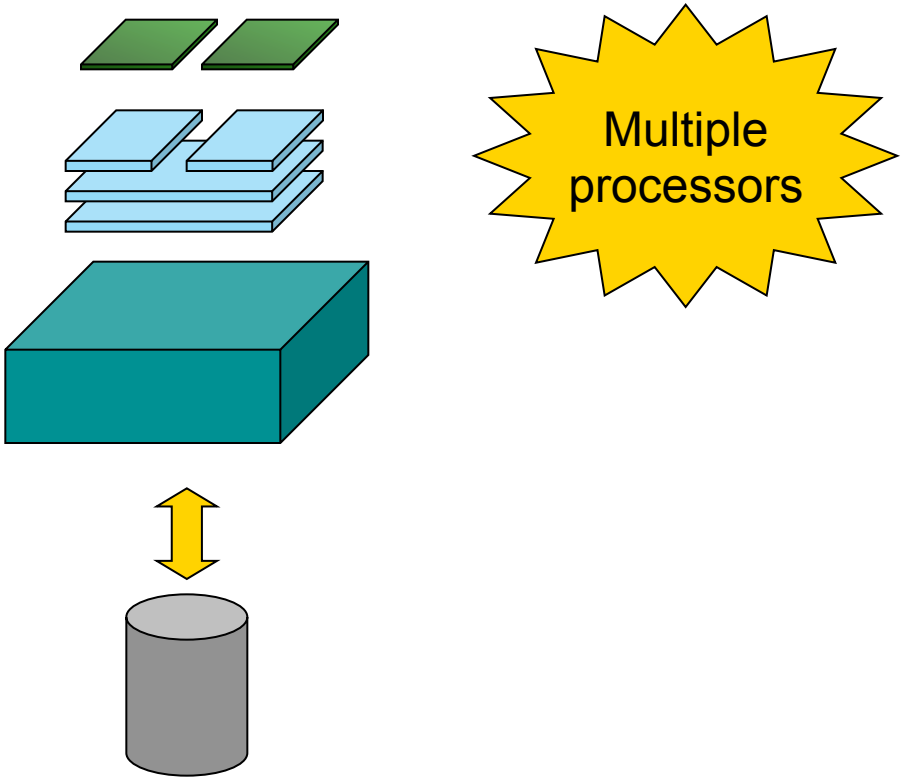
© 2014 IBM Corporation

Hardware Architecture



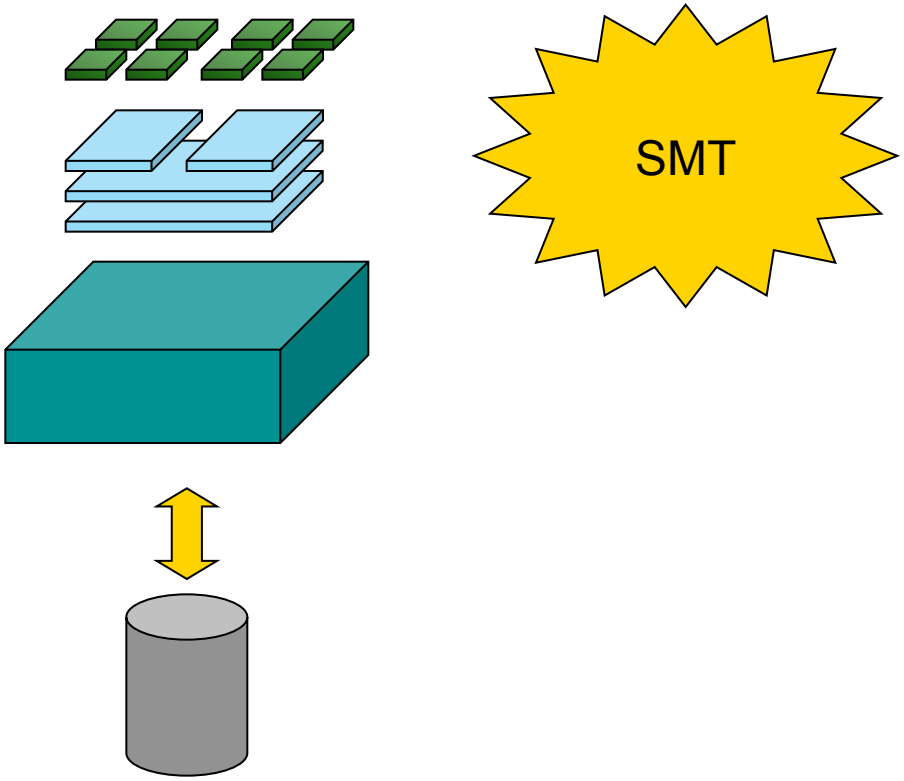
© 2014 IBM Corporation

Hardware Architecture



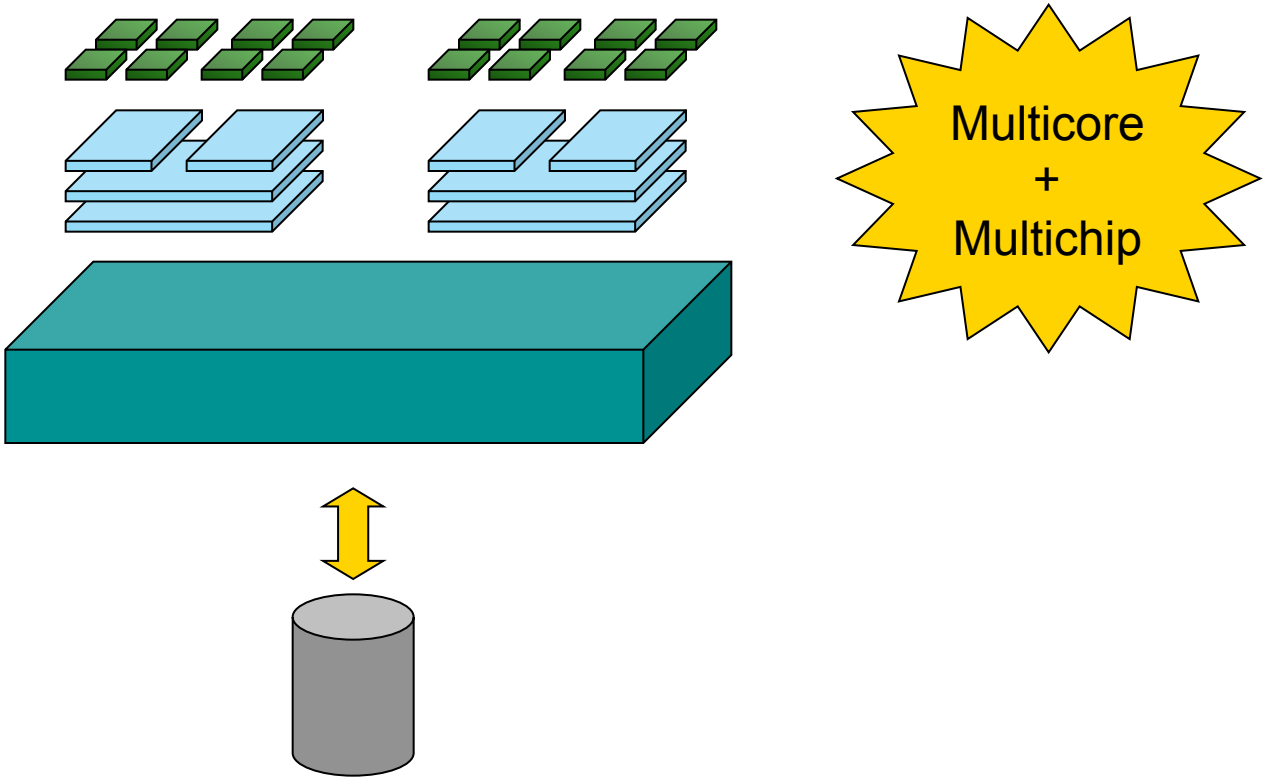
© 2014 IBM Corporation

Hardware Architecture



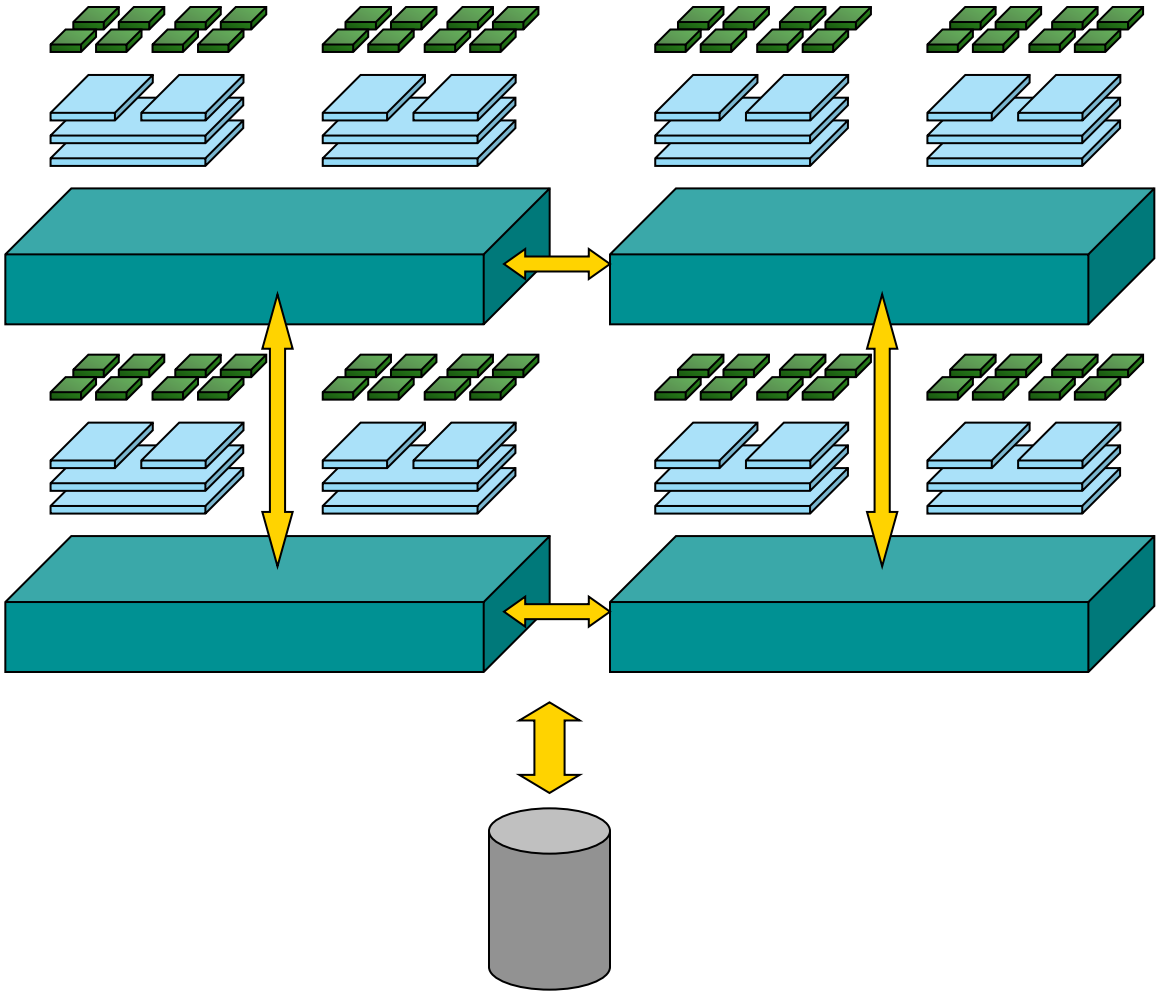
© 2014 IBM Corporation

Hardware Architecture



© 2014 IBM Corporation

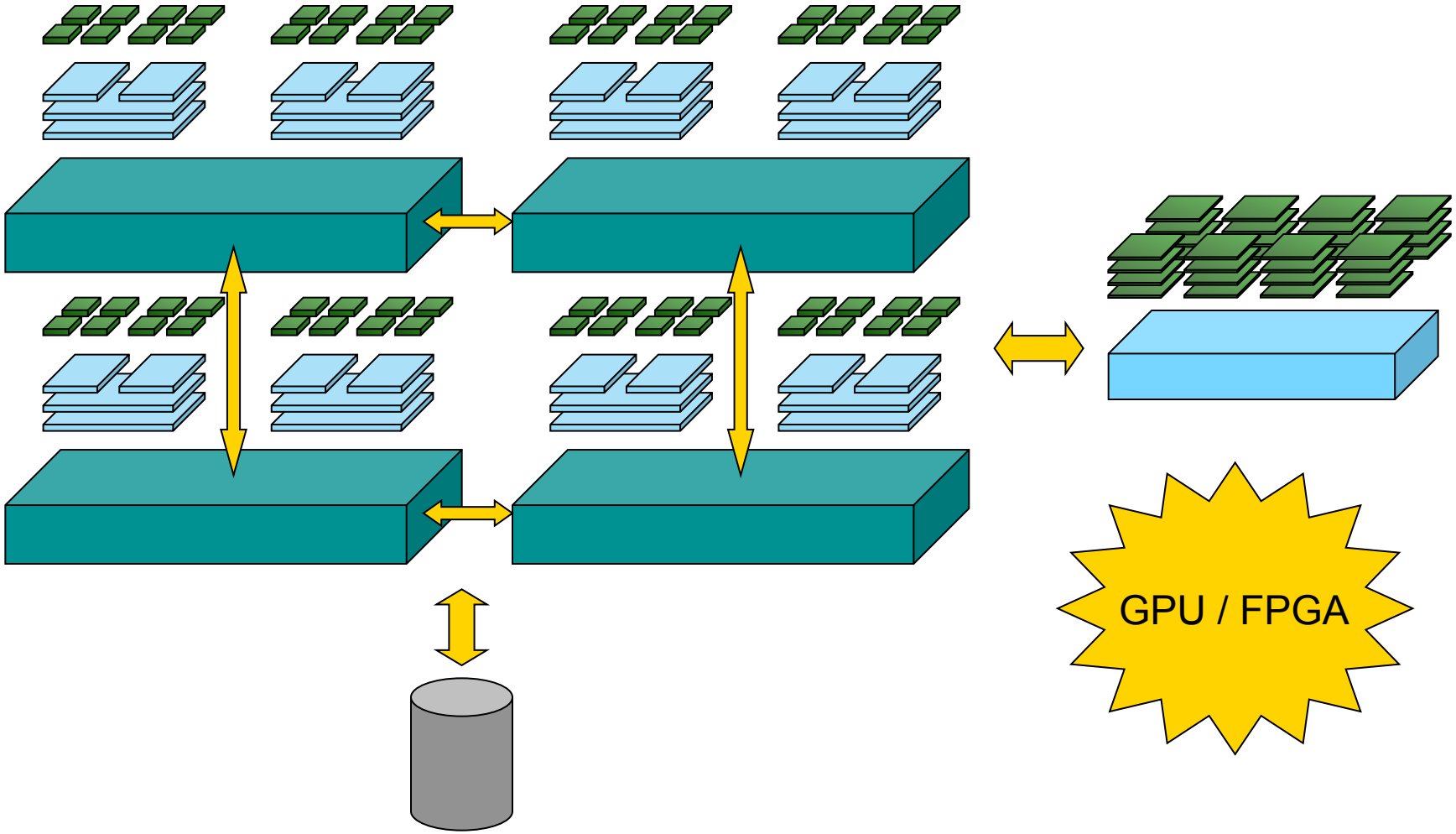
Hardware Architecture





© 2014 IBM Corporation

Hardware Architecture



© 2014 IBM Corporation

CUE JOHN

© 2014 IBM Corporation

Java



- Smalltalk in C's clothing?
- Primitives plus objects
- Concurrency built in – but poorly understood

© 2014 IBM Corporation

More Java Distinctives



Buzzword Compliant!

- Portable “simple” virtual machine
  - –Close enough to the HW
- Secure code load & access control
- Managed pointers and heap (GC)
  - –Data structure integrity
  - –Generous doses of reflection
- Thread friendly
- Just a few non-objects (int, int[])
- Disciplined native interconnect

© 2014 IBM Corporation

Java



"The solutions of today
are the problems of tomorrow."

— Brian Goetz

© 2014 IBM Corporation

JNI: Opening the box



- One of Java's "secret sauces"
- A powerful interop story
- Preserves the freedom of the JVM
- Intended for restricted use
  - Base class libraries
  - Platform interfaces (e.g. AWT / SWT)

© 2014 IBM Corporation

JNI: A victim of its own success?

- Suddenly Java became the universal software adaptor
- Abstraction is great, but it doesn't come cheap



Free Art and Technology [F.A.T.] Lab and Sy-Lab.
"The Free Universal Construction Kit." *Fffff.at*, 20 March 2012. <
http://fffff.at/free-universal-construction-kit>.

© 2014 IBM Corporation

Unsafe: The Anti-JNI?



- Turn the box inside-out
  - No abstractions
  - No protections
  - No documentation
  - Fast … but dangerous
- Intended for restricted use
  - Base class libraries (e.g. NIO, reflect)

© 2014 IBM Corporation

Better JNI: Easier, safer, faster

- FFI more flexible with JNR and on-demand stub spinning
- New array APIs with native implementations for off-heap
- Value types correspond to flattened array elements
- Smart foreign pointers driven by programmable layouts

© 2014 IBM Corporation

What's in a smart pointer?

- (early prototypes) address expression plus type and scope information
- Long addr – either virtual address or intra-object offset
- Object baseObject – either null or the containing managed object
- Layout<T> layout – metadata that protects and controls access
- Optionally, additional management data, to help GC track the parts

© 2014 IBM Corporation

Dereferencing a smart pointer

- p.val() ➔ p.layout.val(p.baseObject, p.addr) ➔ …varhandle/Unsafe stuff
- p.put(x) ➔ p.layout.put(p.baseObject, p.addr, x) ➔ …etc.
- Prototyped as a value-based class; should be a value type.

Details: http://hg.openjdk.java.net/sumatra/sumatra-dev/scratch/file/tip/src/org/openjdk/sumatra/data/prototype/Location.java

© 2014 IBM Corporation

Layout = encapsulating metadata for locations

- Int size, alignment – basic information
- Class<T> cls – type stored at location
- abstract T val(base, addr)
- Should be an *object* type; open-ended (but trusted)
  - Many kinds: C struct, C array, Java array, Fortran array, protocol …
  - Composites (structs, tuples), other aggregates

Details: http://hg.openjdk.java.net/sumatra/sumatra-dev/scratch/file/tip/src/org/openjdk/sumatra/data/prototype/Layout.java

© 2014 IBM Corporation

Smart means flat

- Smart location and layout types are necessary to express flatness
- Many degrees of freedom in layout supports right-sized, well-aligned data

© 2014 IBM Corporation

Where does metadata come from?

- Ultimately, layout info is from language specs, header files, IDL, etc.
- Need good workflows or tools for extracting this into Layout objects
- Perhaps also “little language” and/or “meta data protocol” (cf. MOP)
- Special need: C/C++ header file parser, interface extractor!

© 2014 IBM Corporation

The importance of being Flat

- Memory hardware is not really random-access
- Baked-in preference to ***locally*** sequential access
  - –A result of long co-evolution between HW and SW
  - –Back to mag-tape algos? (Knuth, *Art of Programming*)
- Extra Java indirections and headers interrupt the HW's flow
  - –GC can help, but is not a cure all
  - –And sometimes it randomizes linear access patterns!

© 2014 IBM Corporation

Java vs. Big Data

- 32 bits is too small (no longer “effectively infinite”)
  - Corollary: Long-based indexing of collections
- Can’t afford copying (Java <=> native); zero-copy wins
  - Too big, too slow — the mountain won’t come to us
  - Not all memory created equal (NUMA, GPU)
- Scale matters: terabytes should be chunky, kilobytes flat

© 2014 IBM Corporation

Concurrency management tactics

- Prime directive: Avoid races
- Thread confinement is safe but hard to prove.
- Immutable data structures are safe; need more JVM support.
- Pointers cut both ways!
  - –Nice for atomic updates (think tree-maps)
  - –Arrays can be subdivided by dead reckoning of addresses
- There's also good old monitor-based exclusion and volatiles.
- Future HW might also help with (small scale) transactions
  - –HTM = 2CAS or 3CAS

© 2014 IBM Corporation

For the record: Streaming memory shapes

- Array (flat data) has an episodic life cycle

- Read-only (array-like inputs)
- Scratch (array-like accumulators; better be thread-confined)
- Append-only (buffers for array-like outputs)

- APIs must reflect these bulk-level patterns
  - –Single-element access is not enough

© 2014 IBM Corporation

Java vs. Java — coping with momentum

- Try to predict where HW and languages are going
  - –Hard to guess right 100%, but must attempt it
  - –Also must not leave existing user bases behind
- Source code hints: Short-term answer, long-term cancer.
  - –Think twice before you @annotate
- Avoid optimizations with a best-before date
  - –Can't have another `register` keyword

© 2014 IBM Corporation

The Road(s) Ahead

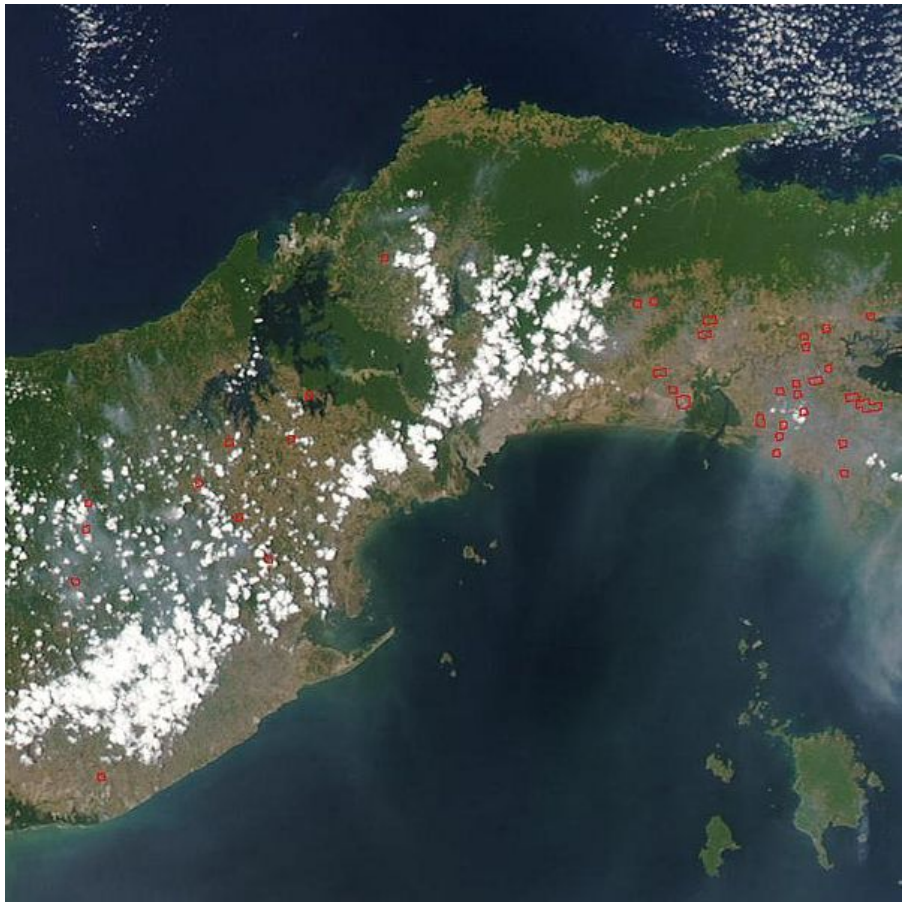
- No single solution will ever cover all the concerns
- Need to clearly delineate the problem spaces
- … in fact, first we have to clearly define the problems!



© 2014 IBM Corporation

Project Panama: Bridging the gap

- Problem space:
  - Zero-copy data access
  - Interoperability
  - FFI
  - Array evolution
- Manipulate data in a way that is:
  - Safe / Secure
  - Approachable
  - Fast (aka JIT-transparent)



http://commons.wikimedia.org/wiki/File:Panama.A2003087.1850.250m.jpg
Cropped from: http://visibleearth.nasa.gov/view.php?id=65881

© 2014 IBM Corporation

Project Valhalla: *The hall of ~~valor~~ **value***



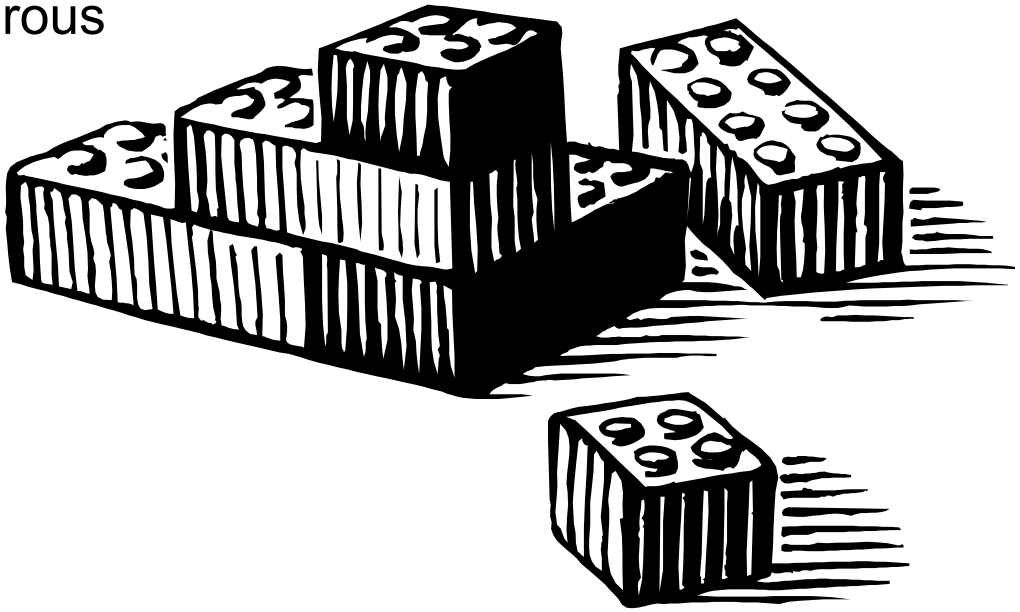
http://commons.wikimedia.org/wiki/File:Walhalla_(1896)_by_Max_Br%C3%BCckner.jpg
"Valhalla" (1896) by Max Brückner

- Problem space:
  - Tuples / extended primitives
  - Truly immutable data
- Flattened data types for:
  - JIT optimizations
  - Reduced overhead
  - Immutability
- Enhanced generics
  - `ArrayList<int>` anyone?

© 2014 IBM Corporation

Future Thinking

- Explicit Java layout?
  - Use cases still need to be explored
  - How to make this powerful …
  - … without making it dangerous



© 2014 IBM Corporation

Parting Thoughts

- The world keeps changing, so either:
  - Invent new languages / runtimes
  - Evolve the old ones
- Remember the lessons of the past
  - But expect future surprises
- Change takes time
- Don’t stop experimenting and agitating for change
  - Exploration and discussion are always the first step

© 2014 IBM Corporation



END

© 2014 IBM Corporation