Execution Environments for Building Dependable Systems

Ben Zorn
Software Design and Implementation
Microsoft Research

The opinions expressed in this presentation are those of the presenter and do not represent the views of Microsoft Corporation

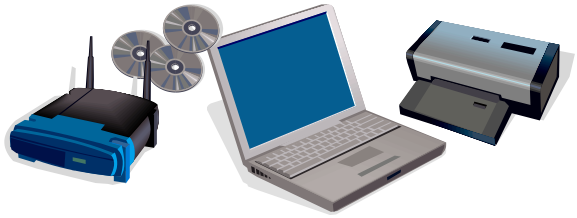
What is a Computer?

pre 1970s (it's hard to find a mainframe picture now)

1970s – minis to micros

1980s – PC revolution

1990s – laptops, networks, CDs



2000s - what's the iconic computer image?

Computing Technology is Changing

- "Intel to ship dual-core Xeon MP in Q1 06" – The Register 3/1/2005
- "Intel is shifting most of its focus in the processor market to dual core CPUs, suggesting that by the end of 2006, better than 75% of the CPUs Intel ships will be multicore processors." ExtremeTech 3/2/2005
- "AMD Details Dual-Core Plans" PCWorld 2/23/2005
- "The Cell processor consists of a general-purpose POWERPC processor core connected to eight special-purpose DSP cores." Ars Technica
- "The first rumor on the actual [Xbox 2] CPU specifications appeared in a February 2004 *Mercury News* story, which reported that the system will have three "IBM-designed 64-bit microprocessors"." Gamespot.com (Hardware) 2/25/2005

Expectations are Changing

- "New worm poses as tsunami relief plea"
  Reuters 3/8/2005
- "First mobile phone virus found in messaging"
  Reuters 3/8/2005
- "New Google tool poses privacy risks"
  AP 10/18/2004
- "LAPD studies facial recognition software"
  AP 12/27/2004
- "Credit card leaks continue at furious pace"
  MSNBC 9/24/2004
- "LexisNexis says 32,000 consumer profiles stolen" Reuters 3/9/2005

Really Dependable Systems

- AED – automated external defibrillator
  - Cheap (< $2000), effective at restarting hearts
  - “Should be as readily available as fire extinguishers”
- Pacemakers
  - “Hackers may target pacemaker technology”
    Portsmouth Herald, 3/16/2005
- Respirocyte* – “post-biological” era
  - 1 micron nanomedical device intended to replace red blood cells
  - 236 times more oxygen / unit volume vs cell
  - 18 billion atoms, onboard nanocomputer

*See ”Respirocytes” by Robert A. Freitas, Jr. (www.KurzweilAI.net)

Insights

- Device form factor and function exploding
  - Special functions with general capabilities
  - Diverse requirements, many need to be dependable
- Applications will drive the system requirements
  - What OS? Execution environment?
- Complexity is the enemy
  - For correctness, security, reliability
  - For performance
  - For agility

Foundations for Future Systems

- What's the equivalent of TCP/IP for software systems?
  - TCP/IP survived 40 years of exponential technology growth (still going strong)
  - Foundation on which great innovation and diversity is based
- Systems need stronger software foundations
  - Tight, well-engineered core
  - Strong, consistent abstraction layers
  - Specifications
  - Multiple independent interoperating implementations (take a page from the IETF handbook)

What's this got to do with MREs?

- Questions
  - How do we build strong software foundations for future applications?
  - What language / MRE / OS is appropriate?
  - What are the relative roles of the MRE and OS?
- Position
  - Pace of technological innovation is gated by the quality of software infrastructure
  - MREs an important part of a future that is racing toward us

Outline

- Motivation
- MREs today
- Challenges for future MRE designs
- Bartok and Singularity
- Thoughts and conclusions

MREs Increasing in Role, Function

- Increasingly dynamic software ecosystem
  - Dynamic libraries
  - Components, plug-ins, applets
- Enhanced programmer productivity
  - High-level (e.g., Visual Basic controls)
  - Less bookkeeping (e.g., GC vs malloc)
- Increasing focus on security, privacy
- Language-level feature integration
  - Threads, security model, isolation model, etc.

Implications of MRE Evolution

- Increasing overlap with OS
  - Example: isolation mechanisms
    - Use OS processes or CLR AppDomains?
  - Projects: KaffeOS – adding OS functions to MRE
  - What is the right boundary?
- Increasing leveraging of metadata
  - Types, reflection, security – expect more in future
  - More data at runtime sustainable?
- Increasing use in new domains
  - Systems, real-time, embedded, etc.

Commercial MREs a Huge Success

- Productivity benefits real, measurable
  - Higher-level abstractions available
  - Code reuse via libraries
  - More errors detected statically, dynamically
  - Reduced bookkeeping, programmer effort
- Many performance challenges overcome
  - Increased engineering, tools, programmer understanding
  - Sophisticated optimization, runtime systems
  - Successful integration of managed / unmanaged code
- Challenges remain…

HeadTrax Experience with .Net

- HeadTrax study (Ovidiu Platon, July 2003, see http://gotdotnet.com/)
  - Multi-tier internal MS app manages HR information
  - Client / server - focus on client experience
  - Client configuration: 128 Mb, 1 GHz CPU
- Implementation
  - Client written in C# with .Net Framework 1.1
  - Network interaction via web services and database APIs
  - Security important – strongly signed binaries, encryption
- Measured startup times
    - Cold start 23 seconds, warm start 10 seconds

Improving HeadTrax Performance

- Implemented
  - Made web service calls asynchronous
  - Cache data locally
  - Lazy instantiation of proxies
  - Show UI before populating
- Cold **23 -> 10** secs, warm **10 -> 8** secs
- Proposed
  - Merge assemblies, DLLs
  - Merge threads, use thread pool

SAP Experience with Java

- "Using VEEs for Standard Business Applications"
  - Hans-Christoph Rohland, VP Java Server Technology, SAP AG
  - Presented at IBM Future of VEEs Workshop, Sept. 2004 (see http://www.research.ibm.com/vee04/)
- Evaluated move from ABAP (in-house MRE) to Java for:
  - Portability – but…runtime behavior is platform specific
  - Security – but…resources not protected by security model
  - Performance – but…performance hard to predict, GC doesn't eliminate memory management problems
  - Productivity – but…tool support insufficient, concurrency is hard
- Conclusions
  - Isolation and layering important (OS also addresses)
  - Non-functional aspects should be better specified

Observations

- Some things require time, engineering
  - 10 seconds is still a long time to wait
    - 1500 16+ Kb chunks read from disk at 6 ms / seek
  - Better tools will be built
- Logical and physical organization are at odds
  - E.g., 21 assemblies, 50 DLLs for 1 app
- Some things are more architectural
  - How do we specify non-functional aspects and build systems to those specifications?
  - How do we make concurrency easier?

Outline

- Motivation
- MREs today
- Challenges for future MRE designs
- Bartok and Singularity
- Thoughts and conclusions

Future Directions for MREs

- Call to action: more innovation, experiments, experience needed
- Many important challenges
  - Performance
  - Correct concurrency
  - “Metadata scale” and data locality
  - Error detection and recovery
  - Core architectural issues
    - Modularity, componentization, versioning
  - “Managed code at the bottom” – building an entire system (App + MRE + OS managed)

Modules, Components, Versions

- Modularity – language support still inadequate
  - How to define large-grain decomposition units?
  - Proposals exist (e.g., IBM MJ, partial classes)
  - How to build systems out of such units?
- MREs are currently one-size fits all
  - Are domain-specific MREs valuable, feasible?
    - Beyond J2EE, J2SE, J2ME
  - What mechanisms are necessary to enable?
- Versioning is a critical part of solution
  - How many components in an MRE?
  - Can they be individually up-leveled?
  - How does this look to an application?

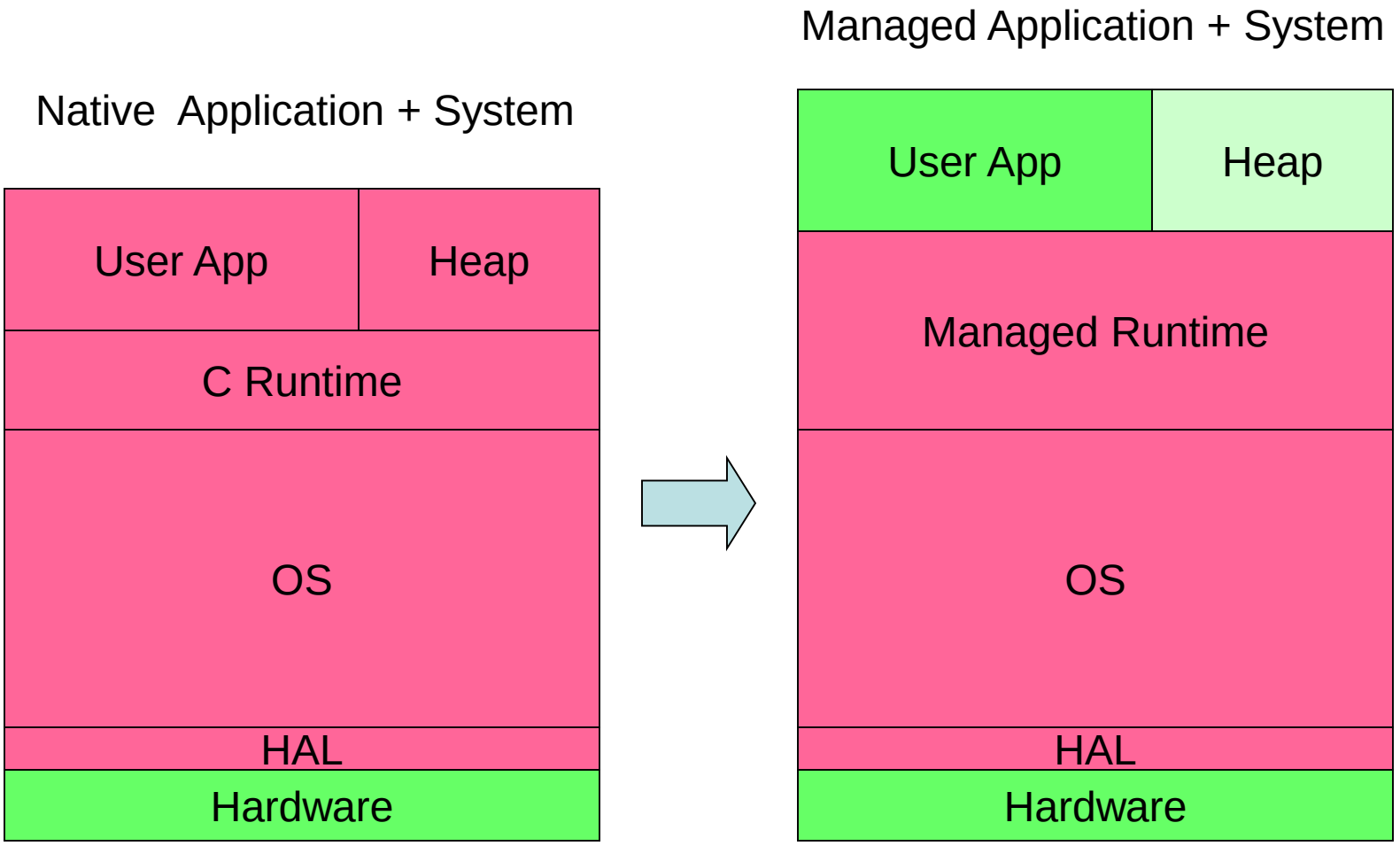
“Managed Code at the Bottom”

- All-managed OS / MRE will be necessary
- Keys to building successful systems
  - GC in the kernel
    - Performance, accounting, integration
    - Encouraging research results
  - Type safety in system code (e.g., GC)
    - Typed-assembly language for runtimes
  - Meeting hard resource constraints
    - Space, real-time, hardened to failure
  - Design with compiler / runtime optimization in mind

Outline

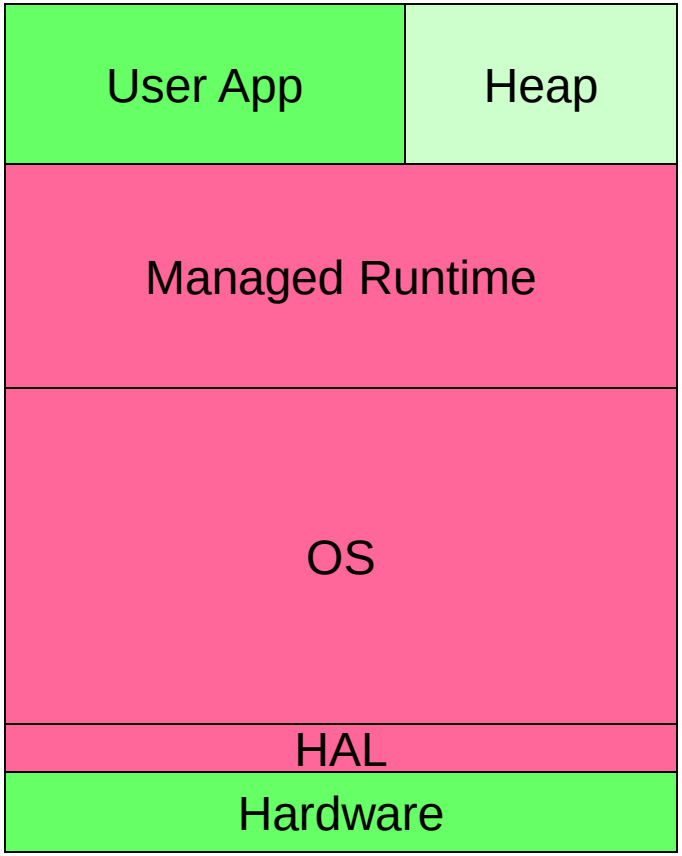
- Motivation
- MREs today
- Challenges for future MRE designs
- Bartok and Singularity
- Thoughts and conclusions

The Evolution of Agile, Dependable Systems

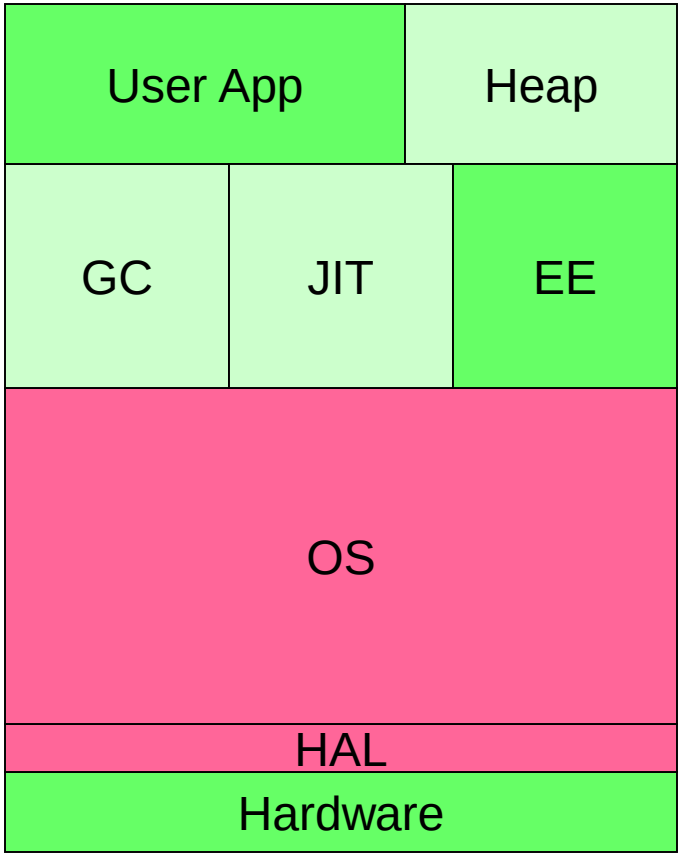


Modular, Type-safe MREs

Managed Application + System



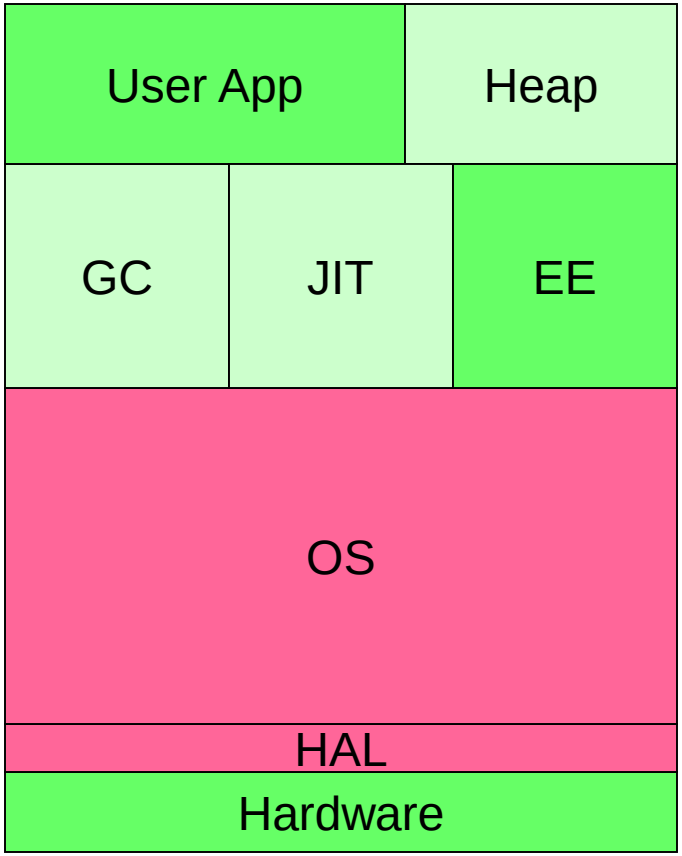
Managed Application + Research MRE



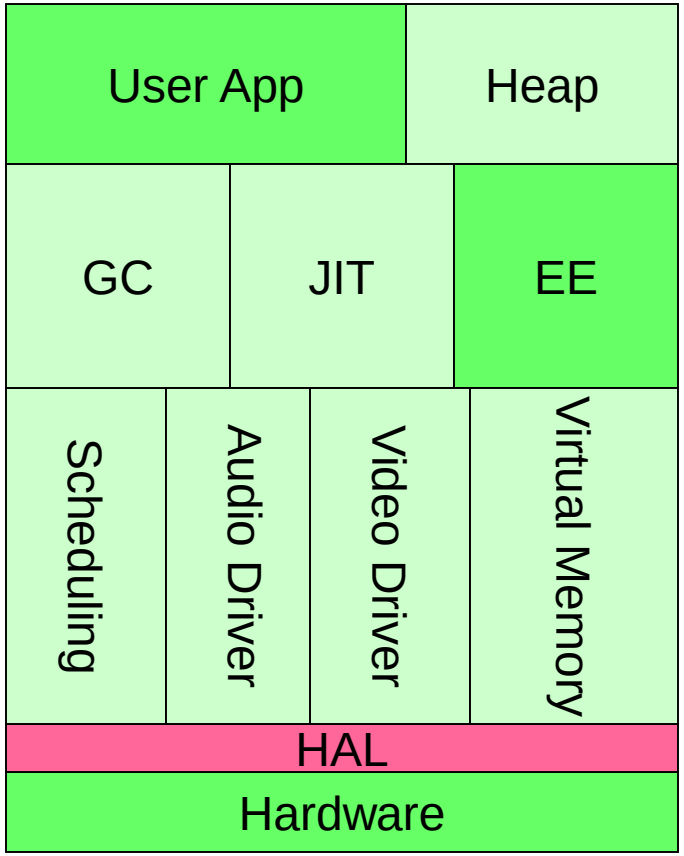
Examples: JMTk, Jikes RVM, ORP

Managed App + MRE + OS

Managed Application +
Research MRE

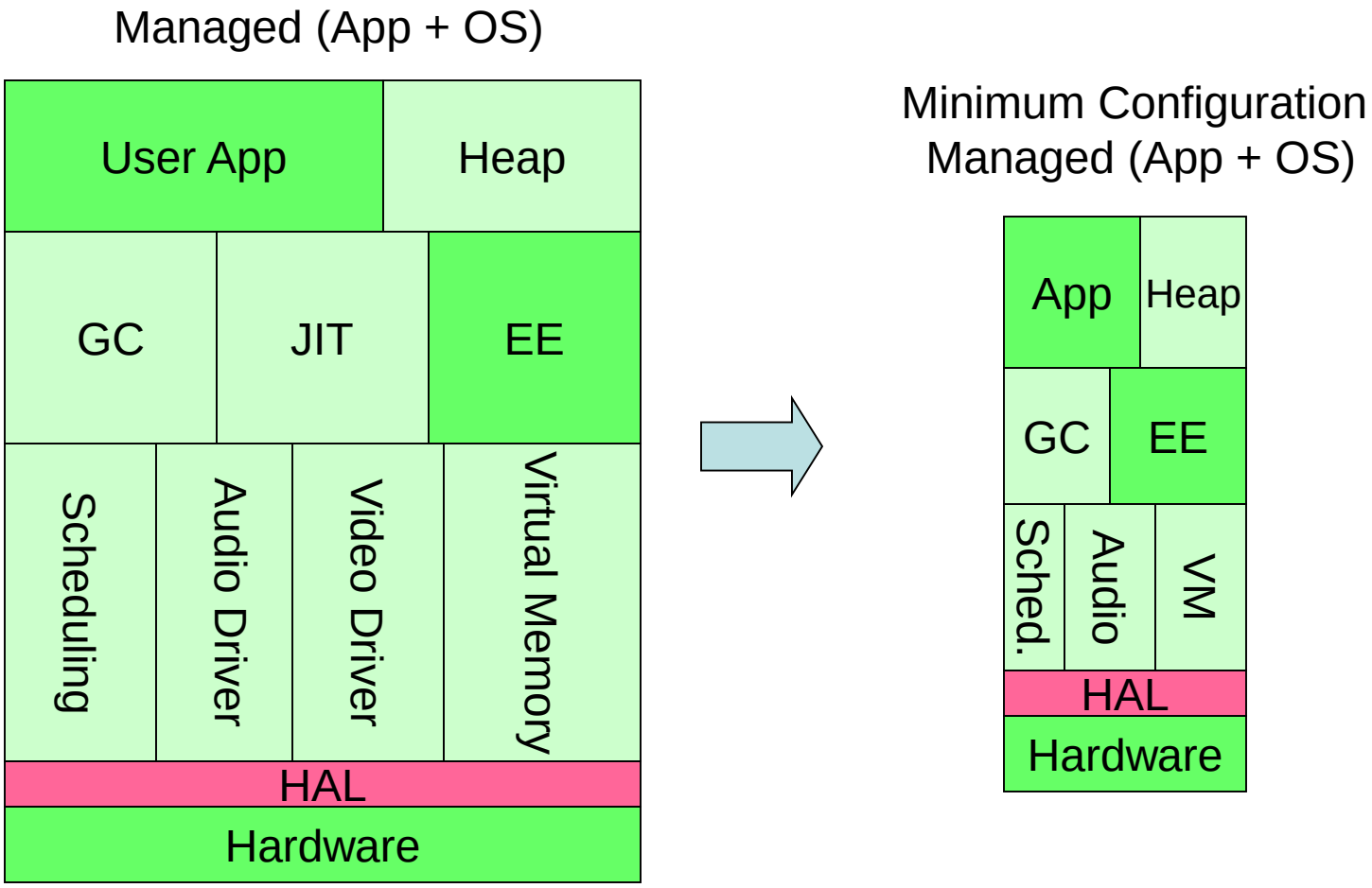


Managed (App + OS)



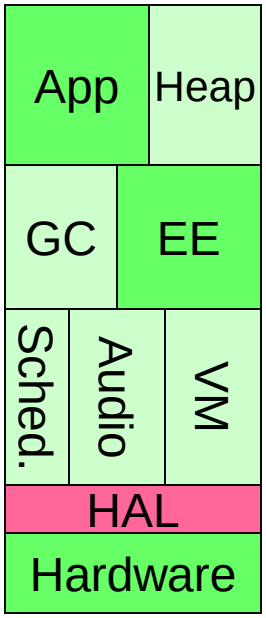
Examples: Singularity, SPIN

Focus on Configurability

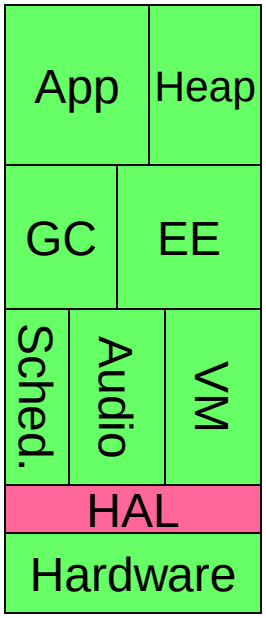


Building Better Abstractions

Minimum Configuration
Managed (App + OS)



Minimum Configuration
Managed (App + OS)
with enhanced abstractions,
tools, checking



Bartok

- What
  - Compiler / runtime system for reduced CLI
  - Support managed-code at the bottom, written in C#
  - Exploring impact of dynamism (reflection, loading, etc.)
  - Developed by ACT group at MSR (Tarditi)
- Research focus areas
  - Optimizations for OO, systems
  - Real-time garbage collection
  - Type-safe abstractions (TAL, well-typed runtimes)
  - Compiler / runtime / OS coupling

Typed Intermediate Languages

- Big picture
  - How much of an MRE implementation can be type-safe?
    - Even the grungy stuff, type-tests, vtables, GC, etc.
  - How about type-safety of MRE implementation *combined* with application code?
    - Check that generated code and meta-data satisfy MRE invariants
    - Optimizations that combine MRE + application code
      - Examples: Inlining type tests, optimizing virtual dispatch
  - How to do this?

Typed intermediate languages

TIL for OO languages: POPL ’05 (Chen and Tarditi)

- Type system:
  - Preserve class names in low-level code
    - Class names are precise
    - Represent objects of a class, not its subclasses
  - Add record types for object layout
  - Allow coercions between objects and records
  - Use class names as bounds in existential types
- Can typecheck low-level code for standard implementation techniques for:
  - Type test, virtual dispatch, interface calls, array covariance checks
  - Including some optimized versions
- Formal semantics, proof of correctness
- Type system ideas could be applied to the source level

Heap Analysis

- Heap analysis not amenable to pure static techniques
- Empirical data indicates program heap is simple
  - Small fraction of heap is actively modified
  - Heap structure is simple
  - Many invariants that are never explicitly stated
- Leverage this to build sound heap abstractions
- Canonical heap representation to combine information from multiple program runs
- Hybrid static-dynamic analyses for soundness & scalability

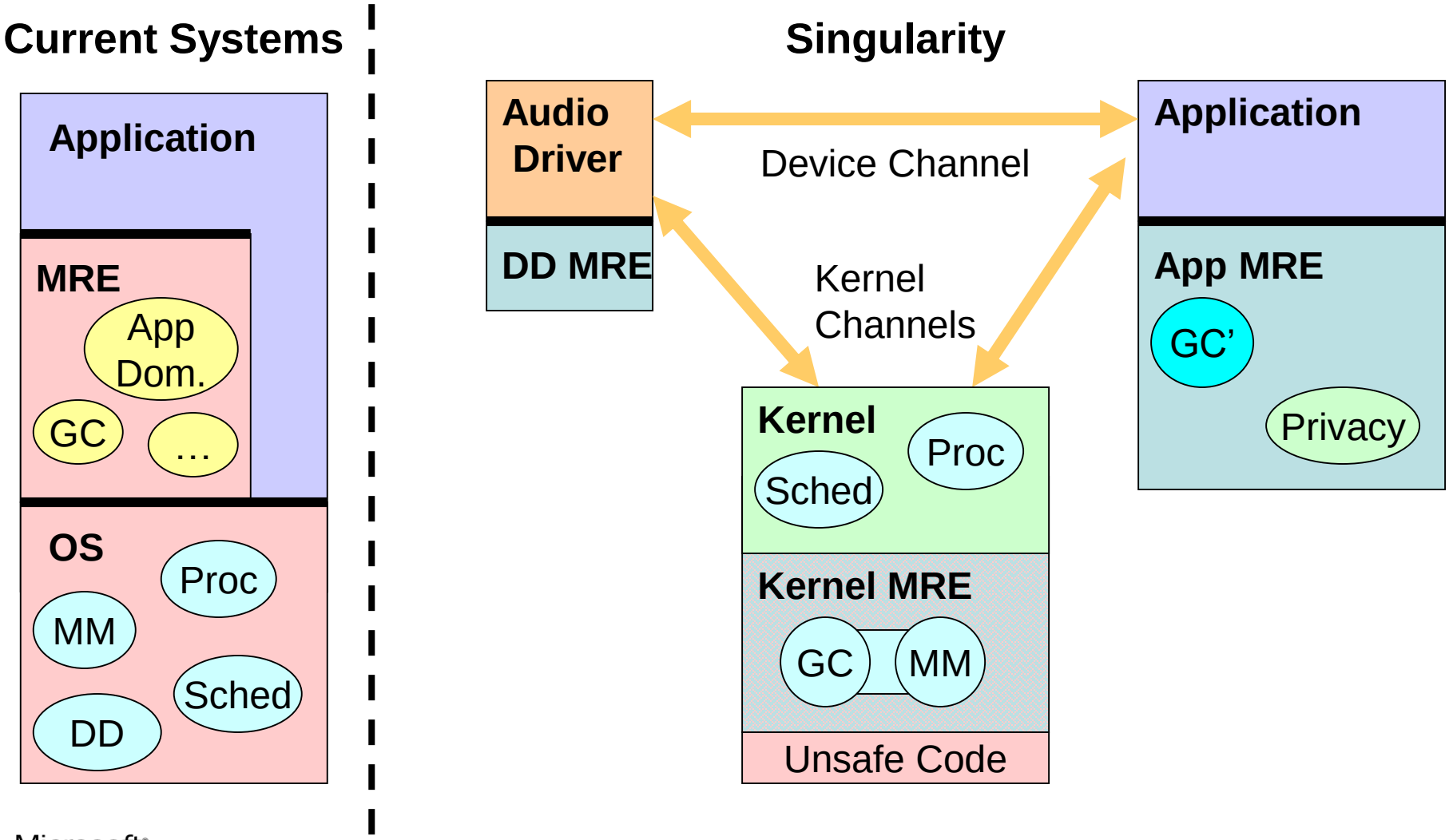
Singularity

- What
  - Multi-group MSR project led by Galen Hunt and Jim Larus
  - New OS design and implementation from ground up
  - Central focus on high dependability
  - Leverages / extends Bartok compiler and runtime
- Design Principles
  - Type-safe (managed) code everywhere
  - Isolate components as much as possible
  - Design for analysis as early as possible
    - Design informed by availability of software analysis tools
  - Willing to trade performance for correctness

Singularity Architectural Elements

- Strong isolation between processes
  - No shared data, communicate via channels
- No dynamic loading, extend via new process
  - Closed world model facilitates checking, customization
- MREs customized on a per-process basis
  - Device drivers, apps, kernel have different MREs
- Checking tools go beyond type-safety
  - Specify, check process interactions
  - Add pre/post conditions (Spec#)
- Reason about the system as a whole
  - Configuration as first-class abstraction
  - Entire system is a self-describing artifact, enabling static inspection and analysis

Vision for OS / MRE Integration



Outline

- Motivation
- MREs today
- Challenges for future EE designs
- Singularity
- Thoughts and conclusions

Summary

- Applications, expectations rapidly changing
  - Software has to keep pace
- MREs are a central part of long-term solution
- Big challenges remain for future designs
  - Core architectural questions need answers
- Related MSR efforts
  - Defining stronger abstractions
    - For language, IL, runtime, heap
  - Bartok - compiler and runtime system
  - Singularity - OS / MRE co-design

Future Investments

- Troubling CISE statistics
  - NSF proposal success rates falling
    - 36% in 1994 to 16% in 2004, some programs much lower
  - Underinvestment in infrastructure
- Software infrastructure research requires increasingly large investment
  - How big before an OS, MRE is “real”
  - Universities, companies need to collaborate
  - MS Phoenix compiler and tools infrastructure is one example
  - Failure to invest, experiment has significant long-term impact

More Information

- Advanced Compiler Technology / Bartok
  - http://research.microsoft.com/act/
- Runtime Analysis and Design
  - http://research.microsoft.com/rad/
- Singularity
  - http://research.microsoft.com/os/singularity/
- Spec#
  - http://research.microsoft.com/projects/specsharp/