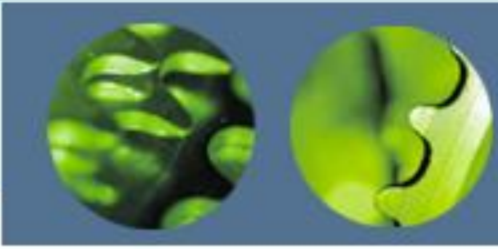
Exploring new OS Research through Singularity

"... it is impossible to predict how a singularity will affect objects in its causal future."

- NCSA Cyberia Glossary

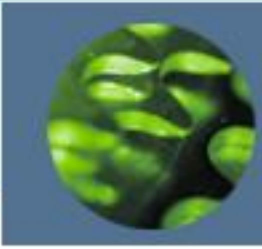
Galen Hunt

Microsoft Research



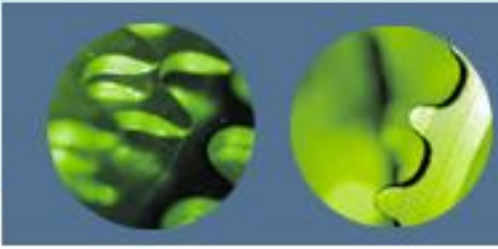
What is OS Research?

- My definition:
  - – Research into the base abstractions to enable computation
  - – Research into implementations for those abstractions (the OS).
- OS Research from 1970 is insufficient today.
  - – OS has adapted to some hardware changes (SMP, Net, etc.)
    - But not the proliferation of cores (GPUs, NICs, etc.)
  - – OS hasn’t adapted to software changes:
    - 1970 Unix Application vs. Windows Application.
  - – OS hasn’t adapted to security changes.
    - Web-based code, faulty device drivers, utility computing, etc.
- Processes, threads, file systems, etc. are probably still relevant.



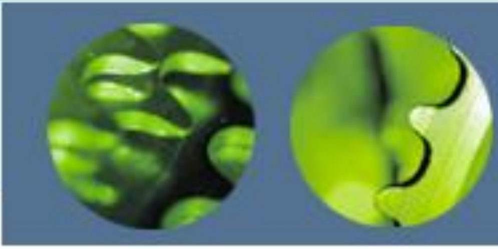
Singularity

- New Research OS
  - Platform for new OS research agenda
  - Common laboratory for the NP cross-group research project into software development w/ PPRC.
- Managed code all the way down.
  - Written in SpeC#
    - Superset of C# (formerly A#)
    - Pre and Post Conditions, Invariants, etc.
  - Bartok runtime (for MSIL) on bare metal.



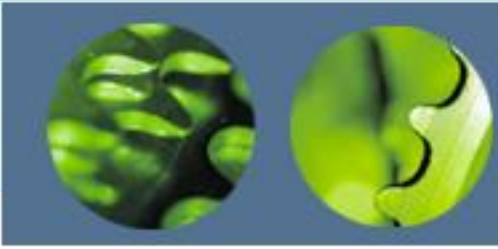
OS Research Agenda

1. Adopt an abstract instruction set (Safe MSIL) as the system binary interface.
2. Add OS abstractions for Ubiquitous Metadata
3. Rebuild the OS security model from top to bottom.
4. Reengineer the OS Development Process
5. Reexamine other OS areas changed by 1-4.



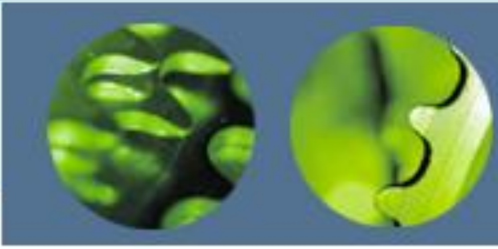
Abstract Instruction Set

- Safe MSIL is the universal OS binary interface
  - Drivers, applications, etc., all arrive as Safe MSIL.
  - Language Safety and Isolation
  - ISA neutral and Structured Metadata
- OS controls translation to native code.
  - Generate qualitatively different code from source MSIL
    - Translate for security vs. performance:
      - buffer overflows, stack walking, NX, SFI, etc.
    - Ubiquitous on demand instrumentation:
      - asserts, profiling, debugging, Vulcan, etc.
  - Rethink traditional hardware security boundaries.
    - Inline privileged operations into non-kernel code.



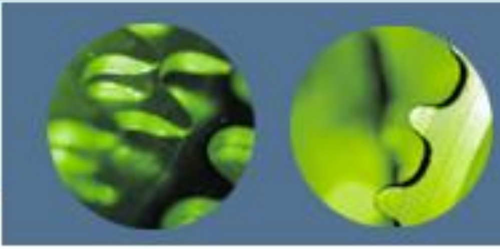
Ubiquitous Metadata

- Make metadata a first class, protected OS feature.
  - Hang extensible metadata off all named OS entities.
    - processes, threads, handles, files, security principals, etc.
  - Strong typing and schemas for OS interfaces
    - registry, ioctls, /proc, etc.
  - Unify OS metadata APIs
    - for enumeration, change notification, security, etc.
- Add OS abstractions to bridge metadata gaps.
  - (Program → Process) is like (Class → Instance)
- Enable complete static analysis of OS *and* applications
  - at compile and deployment time
  - for verification, optimization, versioning, etc.



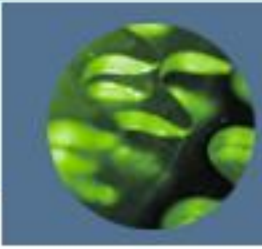
Rebuild
The OS Security Model

- Remove drivers from the TCB
  - Reify driver/service trust relationships.
  - Hardware to ensure memory safety (DMA “MMU”).
- Remove Administrator from the TCB
  - Reify user trust relationships.
  - Orthogonal Management
    - Kernel (OS Vendor) as Trusted Third Party
  - What is “complete” solution for NGSCB scenarios?
- Simplify security management
  - Deep isolation mechanisms.
  - Eliminate partial redundancies between OS and Runtime.



Reengineer the OS Development Process

- Software 2010 Grand Challenge
  - “Routine to build secure, reliable, and maintainable applications & systems”
- NP (New Project with Name Pending):
  - Cross group research project
    - OS, SWIG, SBT, FSE, ACT, etc.
  - Explore co-evolution of OS and tools.
    - Structure OS code to meet tools half way.
    - Push tools research into design and deployment phases.
  - Cooperating independent teams
    - focus on own research domain
    - share problems, solutions, people, ideas with other groups



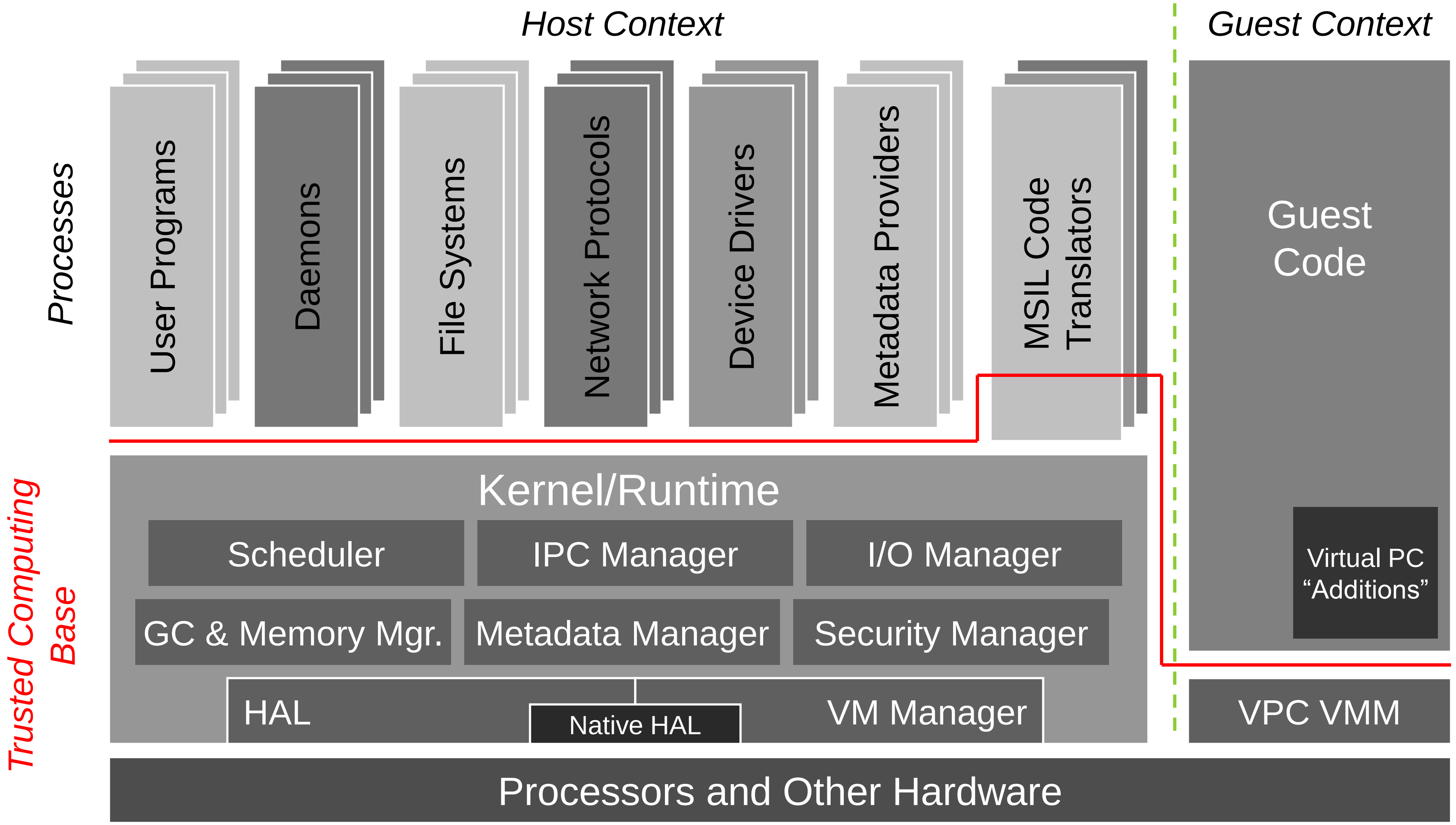
Reexamine Other OS Areas

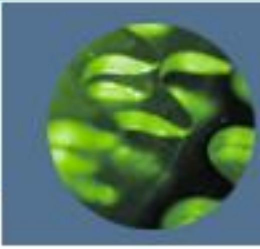
- What should be the boundary between user and kernel code?
- What should a process look like?
- What traditional OS APIs can unified, replaced, or simplified by the Metadata APIs?
- How can we harness the proliferation of CPU cores in modern hardware?
- Should we push transactions deep into the system?

Target Scenario

- eHome digital-convergence
  - Diverse environment with little platform legacy
    - Smart remotes, set top boxes, and media servers.
    - Devices come and go. Storage comes and goes.
    - Wide variety of form factors. One OS.
    - Soft real-time and distributed requirements.
  - Trustworthy, open software platform.
  - Zero on-site human administration.

Architecture





Demo

Demo Description

- What you just saw:
  - Bitmaps (1 per slide) statically linked to Singularity kernel
    - Rendered with safe device drivers.
  - Compiled MSIL running on bare metal.
  - Network boot of Singularity from minidump via PXE.
- What you didn’t see (’cause it ain’t there yet):
  - Threads or a scheduler
  - Interrupts (drivers are polling)
  - Virtual memory, IPC, or a network stack
  - Metadata
- For more info, see http://singularity.