Singularity Project

Yet another operating system?

by Miroslav Rudišin

March 2006

About project

- Who?
  - Microsoft Research OS group
    - Galen Hunt, James Larus, …
- What?
  - building more dependable software platform
- Why?
  - insufficient robustness, reliability, security

Dependability

The notion of dependability, defined as the ***trustworthiness*** *of a computing system which allows reliance to be justifiably placed on the* ***service*** *it delivers*, enables these various concerns to be subsumed within a *single* conceptual *framework*.

Dependability thus includes as special cases such attributes as reliability, availability, safety, security.

The principle

Everything is under control

(should by)

Software-isolated processes

- SIP = closed object space
  - depends on language safety
  - memory independence invariant
- Communication between SIPs
  - sending messages over channels
  - transfers ownership of data
- *Yay, 1979 again? Mainframes?*

Extensibility

- no dynamic code loading
- SIP = encapsulation
  - on failure system frees SIPs resources and notifies communication partners
  - simply way to isolate and discard corrupted data
- no dynamic code generation
- **compile-time reflection**
  - simmilar to macros, aspects, multi-stage programming…

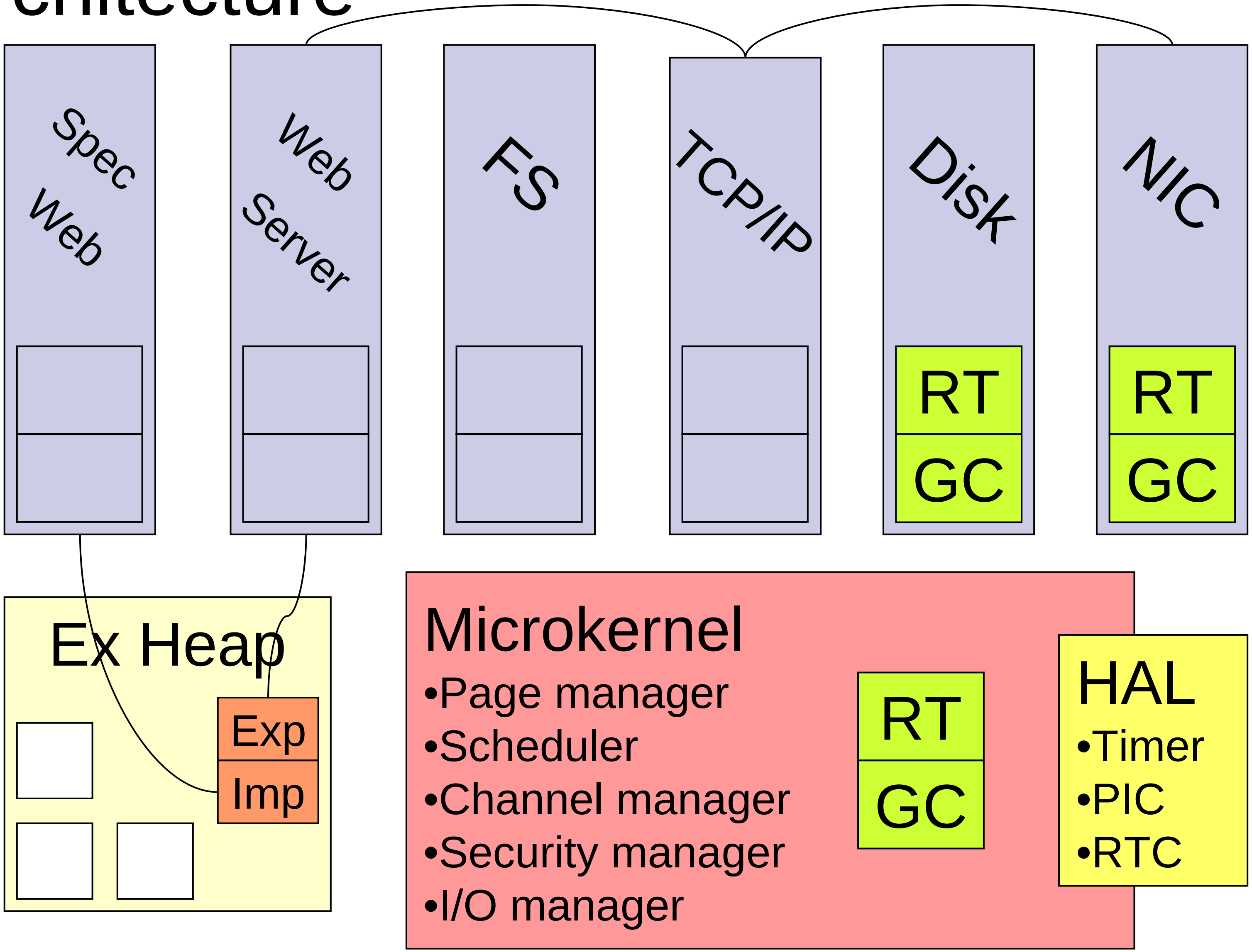
Application abstraction

- application = manifest & resources
- manifest describes desired state
- unified Singularity installer
- system is aware of all relationships

Key contributions

- SIP = verified safe code
- consistent extension model
- run-time verification of execution and communication
- language and compiler
- OS = safe language run-time system

Architecture



Code in Singularity

- C# … Spec# … **Sing#**
  - verification of type-safety, null pointers, exceptions, contract between functions and also protocol-level interactions between components
- Compiled to safe **MSIL**
- Compiled to x86 code by **Bartok** compiler
  - *interpreted/JIT kernel would be insane (Inferno OS, JX)*

- **Trusted code** (HAL, Kernel, parts of RT system, GC)
- assembler, C++, unsafe C#

Kernel

- privileged system component
- own garbage collected object space
- provides Application Binary Interface (ABI)
  - static kernel methods
  - maintains system-wide **state isolation invariant**
  - ABI versioning – *Microsoft.Singularity.V1.Threads*
- compile-time replaceable scheduler
- exports synchronization constructs – **handle table**
  - used for coordination threads within process

Processes

- single virtual address space
- created by allocating memory sufficient for image
- started by kernel by executing trusted startup code
  - stack & memory pages initialization
- **memory independence invariant**
- stack management (per thread)
  - linked stack (non-contigous segments)

- *Everything runs in kernel! No context switches. … False.*

Garbage collection

- prevents memory deallocation errors
- every process is garbage collected (kernel too)
- no universal garbage collector (GC)
- each SIP has its own GC
- 5 selectable GC for each SIP
- independent scheduling of GCs (no cross-object ptrs)
- ABI calls to kernel are executed on thread’s stack
  - solved by marking that area - GC can skip marked frames

Run-time system

- each SIP can have completely different RT system
- allows customization
  - sequential code without threads
  - specific allocation strategies such as preallocation
  - …
- own memory layout, garbage collection algorithms, libraries

- *Can SIP select scheduling policy for its threads? Probably no.*

Exchange Heap

- holds data passed between processes
- reference counts to track usage of **regions**
- process accesses region throught structure **allocation**
- strict ownership that maintains memory isolations
- **tracked resources** – via Sing# abstraction **TRef**
  - □ blocking **Acquire** of ownership, **Release**
- allocating memory on ExHeap, req. **Exchangeable type**

```
R In ExHeap a;              /* exchangeable type */
R * In ExHeap pr;           /* pointer to R */
R [] In ExHeap pv;          /* vector of R */
```

Channels

- SIPs comunicating by sending messages over channels
- bidirectional, behaviourly typed **contract**
- messages are tagged data in Exchange Heap
- send is asynchronous
- receive synchronously blocks
- **ownership invariant**
  - prevents sharing
  - helps static program analysis
  - provides message-passing sematics

Channel contract

- **Sing#**

-
  ```
  contract C1 {
          in message Request(int x) requires x>0;
          out message Reply(int y);
          out message Error();
          state Start: Request?
                  -> (Reply! or Error!)
                  -> Start;
  }
  ```

- C1.NewChannel(**out** Imp, **out** Exp)

Endpoints

- Pair of endpoints: **Imp**, **Exp**

-
  ```
  C1.Imp {
          void SendRequest(int x);
          void RecvReply(out int y);
          void RecvError();
  }

  C1.Exp {
          void RecvRequest(out int x);
          void SendReply(int y);
          void SendError();
  }
  ```

Switch-Receive Statement

```
void M (C1.Imp a, C1.Imp b) {
    switch receive {
        case a.Reply(x) && b.Reply(y):
            Console.WriteLine(“Both replies {0} and{1}”, x, y);
            break;
        case a.Error():
            Console.WriteLine(“Error reply on a”);
            break;
        case b.Error():
            Console.WriteLine(“Error reply on b”);
            break;
        case a.ChannelClosed():
            Console.WriteLine(“Channel a is closed”);
            break;
    }
}
```

Compile-Time Reflection (CTR)

- generators are written in Sing# as transforms
- basic idea … place-holders are expanded by generator
- usage scenario:
  - device driver describes its resource requirements
  - startup code can be generated from this description
- CTR transform may be part of trusted base, so it can generate trusted code

CTR Example

```
transform DriverTransform
where $IoRangeType: IoRange {
    class $DriverCategory: DriverCategoryDeclaration {
        [$IoRangeAttribute(*)]
        $IoRangeType $$ioranges;
        public readonly static $DriverCategory Values;
        generate static $DriverCategory() {
            Values = new $DriverCategory();
        }
        implement private $DriverCategory() {
            IoConfig config = IoConfig.GetConfig();
            Tracing.Log(Tracing.Debug, "Config: {0}", config.ToPrint());
            forall ($cindex = 0; $f in $$ioranges; $cindex++) {
                $f = ($f.$IoRangeType) config.DynamicRanges[$cindex];
            }
        }
    }
}
```

CTR Example (cont.)

```
internal class Sb16Resources: DriverCategoryDeclaration {
    [IoPortRange(0, Default = 0x0220,Length = 0x10)]
    internal readonly IoPortRange basePorts;
    [IoPortRange(1, Default = 0x0380,Length = 0x10)]
    internal readonly IoPortRange gamePorts;
    internal readonly static Sb16Resources Values;
    reflective private Sb16Resources();
}

class SB16Resources {
    …
    static Sb16Resources() { Values = new Sb16Resources(); }
    private SB16Resources() {
        IoConfig config = IoConfig.GetConfig();
        Tracing.Log(Tracing.Debug, "Config: {0}", config.ToPrint());
        basePorts = (IoPortRange) config.DynamicRanges[0];
        gamePorts = (IoPortRange) config.DynamicRanges[1];
    }
}
```

Verification

- three-stage process
  - Sing# - type safety, ownership rules, protocol
  - Singularity verifier on MSIL code
  - (not yet) Back-end compiler produces typed assembly
    - enables run-time checks by operating system
- mostly with static code analysis
- small impact on performance

Singularity System

- I/O System – 3 layers: HAL, I/O manager, device drivers
- HAL: IoPorts, IoDMA, IoIrq, IoMemory
- device drivers binded by theirs manifests (metadata)
- maintains **3 device driver (DD) invariants**:
    - never installs DD conflicting with other DD
    - never starts DD with conflicting or missing resource
    - DD cannot access unspecified resources

Singularity Namespace

- single, uniform namespace for all services
- UNIX-like mountpoints
- examples:

/hardware/devices
/hardware/drivers
/filesystems/ntfs
/tcp/128.0.0.1/80
/fs/foo/bar
/apps/ms/word
/users/fred

Security

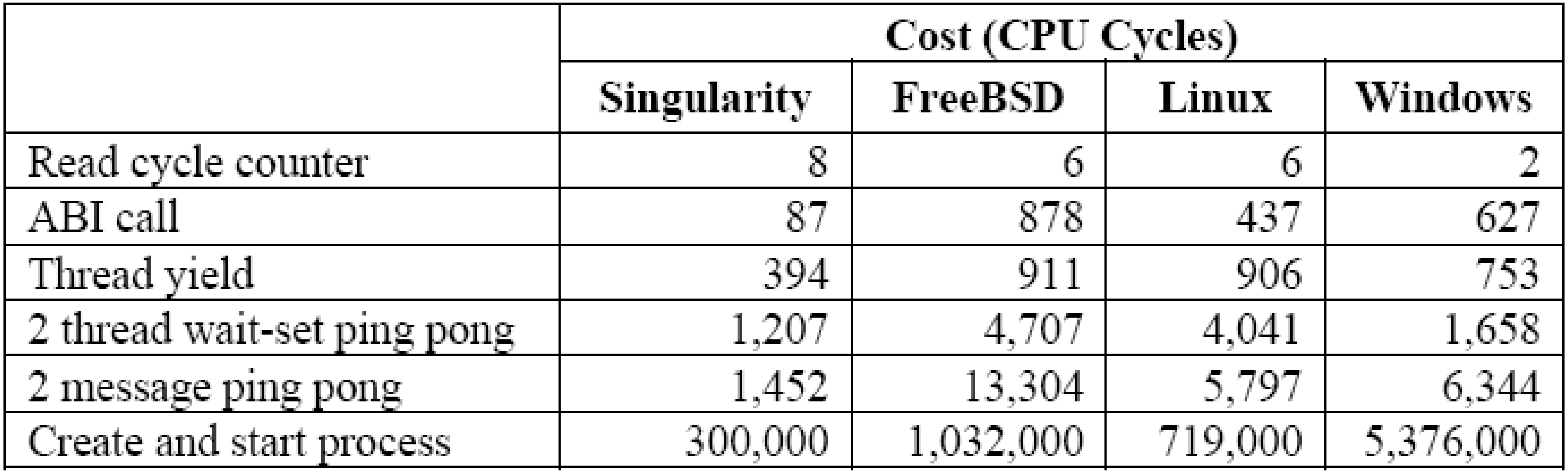
- shape of namespace
- enforcing access by managing channels of application according to its manifest
- discoverable process identity at the peer side of channel
- trace of process execution history
- compound principals:
  - /sys/login @ /users/fred + /apps/ms/word
- access control expresions (ACE)
- restriction on sending some messages on channel
- lending identities

Proposed changes on HW

- memory protection for DMA transfers
  - only unsafe aspect of driver-device interface
- support for segmented stacks
- simpler memory protection for trusted base
  - everything runs in RING0

Performance - microbenchmarks

- AMD Athlon 64 3000+, nForce 4 chipset, 1GB RAM, 7200 SATA, Gigabit NIC
- Cost of basic operations:



	Cost (CPU Cycles)			
	Singularity	**FreeBSD**	**Linux**	**Windows**
Read cycle counter	8	6	6	2
ABI call	87	878	437	627
Thread yield	394	911	906	753
2 thread wait-set ping pong	1,207	4,707	4,041	1,658
2 message ping pong	1,452	13,304	5,797	6,344
Create and start process	300,000	1,032,000	719,000	5,376,000

Disk I/O benchmarks

- random & sequential disk reads and writes
- 1000 operations / 512MB
- Singularity - disk driver SIP, channels – zero copy
- other OSes by system calls
- different blocksizes
- all systems has comparable performance (6% diff)
- Conclusion: Singularity has comparable disk I/O performance

SPECweb99 benchmarks

- Cassini open-source C# web server ported to SIPs
- Singularity: 91 ops/sec
- MS Windows 2003 - IIS: 761 ops/sec
- Singularity – instability under heavy load & FS bottleneck

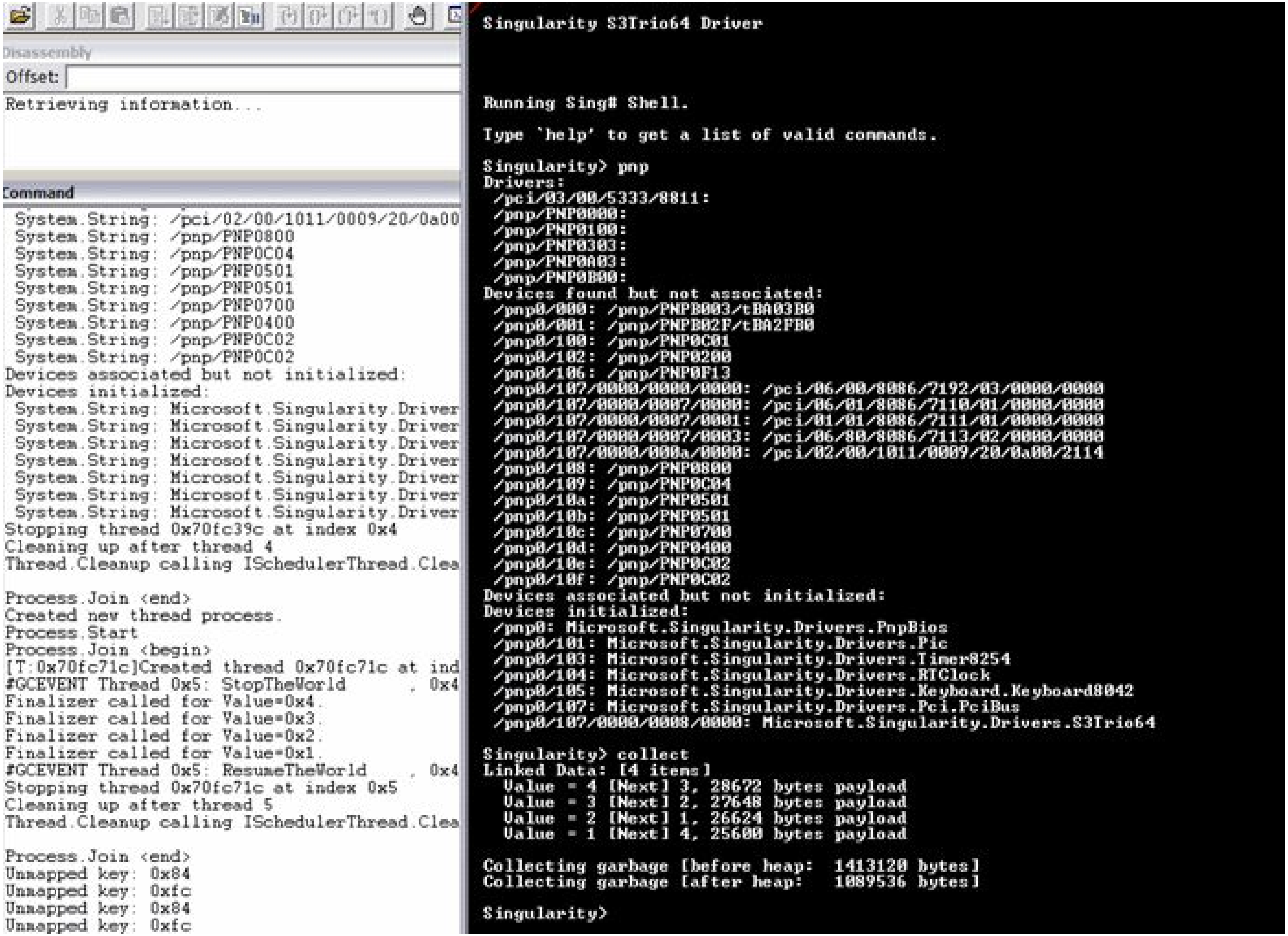
- Response time: Singularity at 23 ops/sec was 322 ms/op

- Network stack does not appear to be bottleneck

Visions & plans

- better performance
- checking for liveness/deadlocks

- specification of actions when component fails
- distributed Singularity
- integration of heterogenous execution environment
  - .NET, EJB, CCM, DCOM, …
- …



```
Retrieving information...
```

```
 System.String: /pci/02/00/1011/0009/20/0a00
 System.String: /pnp/PNP0800
 System.String: /pnp/PNP0C04
 System.String: /pnp/PNP0501
 System.String: /pnp/PNP0501
 System.String: /pnp/PNP0700
 System.String: /pnp/PNP0400
 System.String: /pnp/PNP0C02
 System.String: /pnp/PNP0C02
Devices associated but not initialized:
Devices initialized:
 System.String: Microsoft.Singularity.Driver
 System.String: Microsoft.Singularity.Driver
 System.String: Microsoft.Singularity.Driver
 System.String: Microsoft.Singularity.Driver
 System.String: Microsoft.Singularity.Driver
 System.String: Microsoft.Singularity.Driver
 System.String: Microsoft.Singularity.Driver
Stopping thread 0x70fc39c at index 0x4
Cleaning up after thread 4
Thread.Cleanup calling ISchedulerThread.Clea

Process.Join <end>
Created new thread process.
Process.Start
Process.Join <begin>
[T:0x70fc71c]Created thread 0x70fc71c at ind
#GCEVENT Thread 0x5: StopTheWorld      , 0x4
Finalizer called for Value=0x4.
Finalizer called for Value=0x3.
Finalizer called for Value=0x2.
Finalizer called for Value=0x1.
#GCEVENT Thread 0x5: ResumeTheWorld    , 0x4
Stopping thread 0x70fc71c at index 0x5
Cleaning up after thread 5
Thread.Cleanup calling ISchedulerThread.Clea

Process.Join <end>
Unmapped key: 0x84
Unmapped key: 0xfc
Unmapped key: 0x84
Unmapped key: 0xfc
```

```
Singularity S3Trio64 Driver



Running Sing# Shell.

Type `help' to get a list of valid commands.

Singularity> pnp
Drivers:
 /pci/03/00/5333/8811:
 /pnp/PNP0000:
 /pnp/PNP0100:
 /pnp/PNP0303:
 /pnp/PNP0A03:
 /pnp/PNP0B00:
Devices found but not associated:
 /pnp0/000: /pnp/PNPB003/tBA03B0
 /pnp0/001: /pnp/PNPB02F/tBA2FB0
 /pnp0/100: /pnp/PNP0C01
 /pnp0/102: /pnp/PNP0200
 /pnp0/106: /pnp/PNP0F13
 /pnp0/107/0000/0000/0000: /pci/06/00/8086/7192/03/0000/0000
 /pnp0/107/0000/0007/0000: /pci/06/01/8086/7110/01/0000/0000
 /pnp0/107/0000/0007/0001: /pci/01/01/8086/7111/01/0000/0000
 /pnp0/107/0000/0007/0003: /pci/06/80/8086/7113/02/0000/0000
 /pnp0/107/0000/000a/0000: /pci/02/00/1011/0009/20/0a00/2114
 /pnp0/108: /pnp/PNP0800
 /pnp0/109: /pnp/PNP0C04
 /pnp0/10a: /pnp/PNP0501
 /pnp0/10b: /pnp/PNP0501
 /pnp0/10c: /pnp/PNP0700
 /pnp0/10d: /pnp/PNP0400
 /pnp0/10e: /pnp/PNP0C02
 /pnp0/10f: /pnp/PNP0C02
Devices associated but not initialized:
Devices initialized:
 /pnp0: Microsoft.Singularity.Drivers.PnpBios
 /pnp0/101: Microsoft.Singularity.Drivers.Pic
 /pnp0/103: Microsoft.Singularity.Drivers.Timer8254
 /pnp0/104: Microsoft.Singularity.Drivers.RTClock
 /pnp0/105: Microsoft.Singularity.Drivers.Keyboard.Keyboard8042
 /pnp0/107: Microsoft.Singularity.Drivers.Pci.PciBus
 /pnp0/107/0000/0008/0000: Microsoft.Singularity.Drivers.S3Trio64

Singularity> collect
Linked Data: [4 items]
   Value = 4 [Next] 3, 28672 bytes payload
   Value = 3 [Next] 2, 27648 bytes payload
   Value = 2 [Next] 1, 26624 bytes payload
   Value = 1 [Next] 4, 25600 bytes payload

Collecting garbage [before heap:  1413120 bytes]
Collecting garbage [after heap:   1089536 bytes]

Singularity>
```

The End.

References:

Technical Report (2005-135)

http://research.microsoft.com/os/singularity/

Singularity Revisited (video)

http://channel9.msdn.com/Showpost.aspx?postid=141858

Jeff Darcy's short review of Singularity

http://pl.atyp.us/wordpress/?p=991

Article and discussion on Slashdot

http://slashdot.org/article.pl?sid=05/11/03/1744230