jbox.dk

jbox

Java Server Appliances

Datalogforeningen

November 2002

Michael Ringgaard (mringgaa@csc.com)
Søren Gjesse (sgjesse@csc.com)

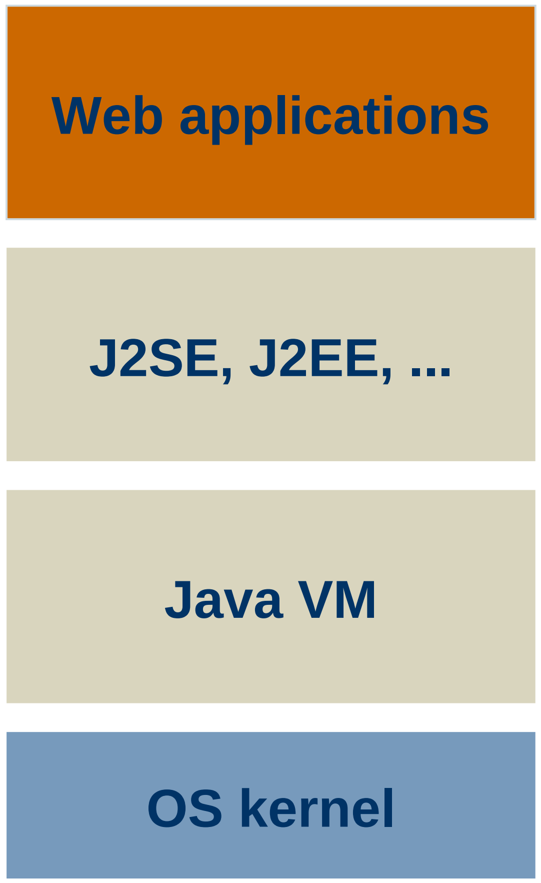
Computer Sciences Corporation

Welcome & Agenda

- Welcome
  - Søren Gjesse, sgjesse@csc.com
    Computer Sciences Corporation
  - Michael Ringgaard, mringgaa@csc.com
    Computer Sciences Corporation
- Agenda
  - What is a jbox?
  - Why would we want a jbox?
  - What can a jbox be used for?
  - How to build a jbox

What is a jbox?

- A server appliance for Java programs
  - Requires only power and a network connection
  - No monitor, keyboard, or mouse
- Built for standard Intel based PC
  - Cheap, simple and powerful
- Runs only one process: the Java VM
  - Specifically the HotSpot Java VM for Windows
  - Relies on a small and efficient kernel
- Transforming application servers to appliances



Reduce System Complexity

- The application complexity is only one part of the overall system complexity
- Goal: Reduce overall system complexity to achieve improvements in:
  - performance
  - flexibility
  - stability
  - security
  - cost

Appliances

- Characteristics of appliances
  - Unpack, connect, use...
  - Can’t rely on experts to operate ..
  - Must require just about zero maintenance
- Would be nice characteristics for an IT solution!
- Servers are most often single-function
- By the way...next generation of home appliances: Broadband router, DHCP, DNS,...

Why a *Java* Appliance?

- Need effective development and execution platform
  - Hardware:
    - Before: Exotic processor/hardware
    - Now: Complies with PC specification
  - Development platform
    - Before: C, C like variant, or assembler
    - Now: OO, VM, garbage collection
- Cost effective
  - Extremely cheap hardware
  - Develop on PC, execute on appliance
  - Wide selection of development environments, tools, utilities...
  - No specialized developers
- *Java is a powerful and rich environment yet simple enough to use in an appliance*

Why not use a standard OS?

- Java is available on most operating systems
  - Most operating systems are candidates

- Contains lots of features not needed for an appliance
- All these features
  - Needs to be updated with latest patches
  - Needs to be monitored
  - Adds security risks
  - Cannot be un-installed

- Is Linux an alternative?
  - Standard distributions have the same problems
  - RedHat uses 100MB+
  - Build your own kernel

Why focus on Intel hardware

- PCs have made Intel hardware cheap
- All kinds of interface components
  - Motherboards with most common interfaces
  - PCI (and/or) ISA cards for almost any purpose
  - Requires drivers to be developed
- Comes in all forms and sizes
  - Embedded (PC104)
  - Cube
  - 1 unit servers
  - Blade servers

What can a jbox be used for?

- What would we like to achieve?
  - Apply the virtues of traditional appliances to IT solutions
  - Apply the effective software development tools, utilities, and methodologies to appliance development
- As a Java server appliance
  - Ideal development environment to develop, deploy and maintain software for appliances
- As a Java cluster node
  - It's better to own 100 appliances than 100 application servers

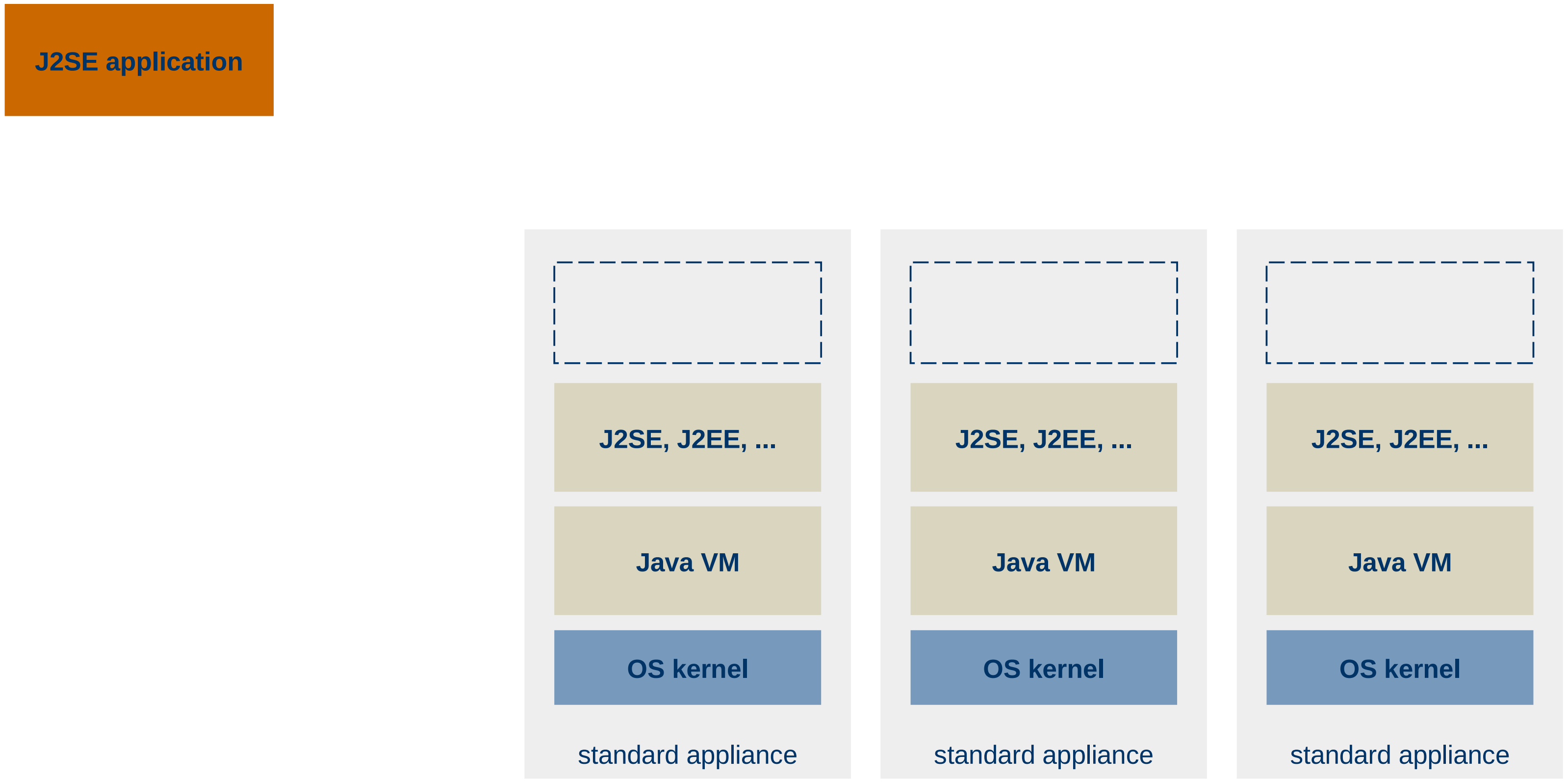
jbox.dk

A Java server appliance

- Single function server ”in-a-box”
  - Burglar alarm
  - DNS
  - Mail server
- Typical form factors
  - Embeded
  - Cube
- Might not need a real harddisk
  - Compact Flash for embeded design
  - SAN/NAS for data storage
- Managed using a browser interface

jbox.dk

Just add water...



Ready in just 5 minutes!!!

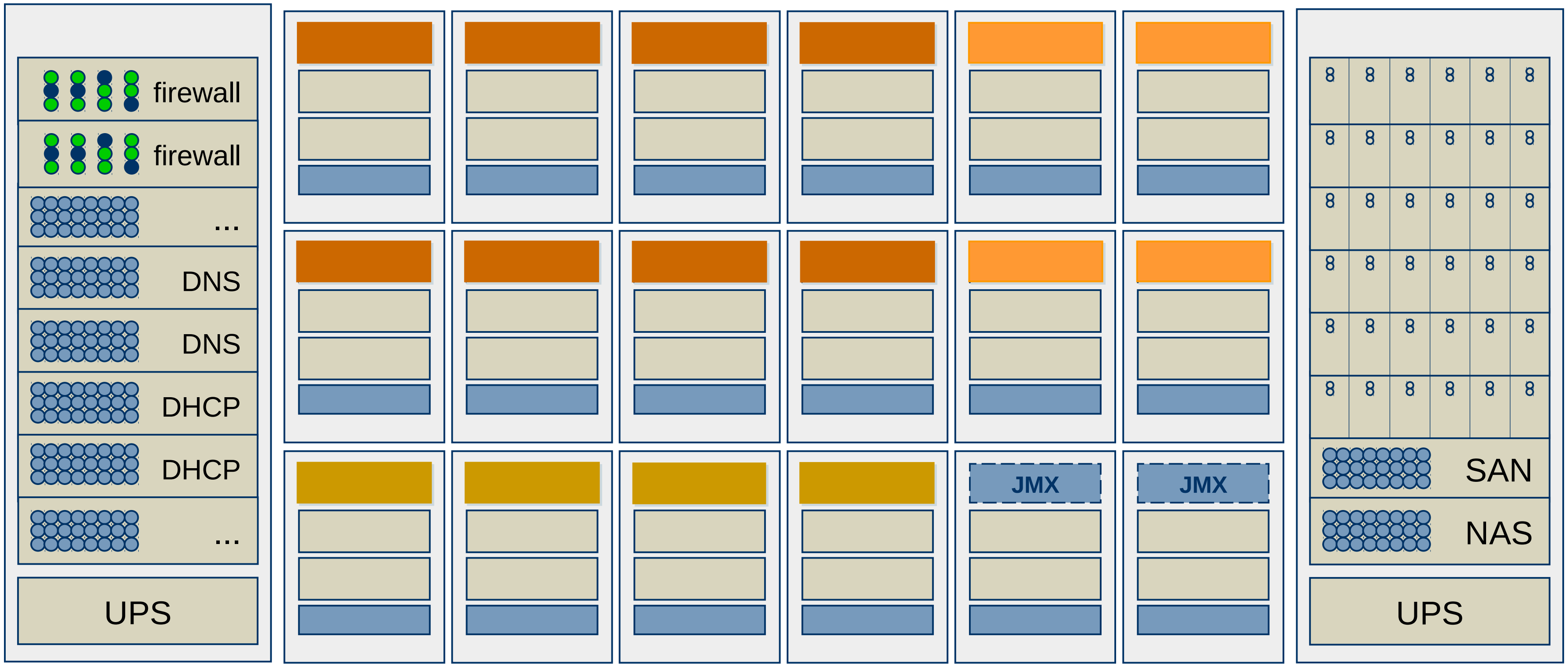
Clusters of Appliances

- Clustering support is a feature of specific J2EE server products
  - Focus on transparency (developer, user)
- Custom designed distributed architectures
  - J2EE +
  - Jini, JavaSpaces
  - P2P, JXTA, ...

Appliance Management

- Standalone appliances can be managed using browser interface
- Most J2EE servers has built-in web based management consoles
- Appliance clusters requires special attention on deployment and configuration issues
  - How do you deploy applications to many nodes
  - Centralized application and configuration repository (JMX).
- *Manage applications, not servers*

Application Clusters



jbox.dk

jbox – Java Server Appliances

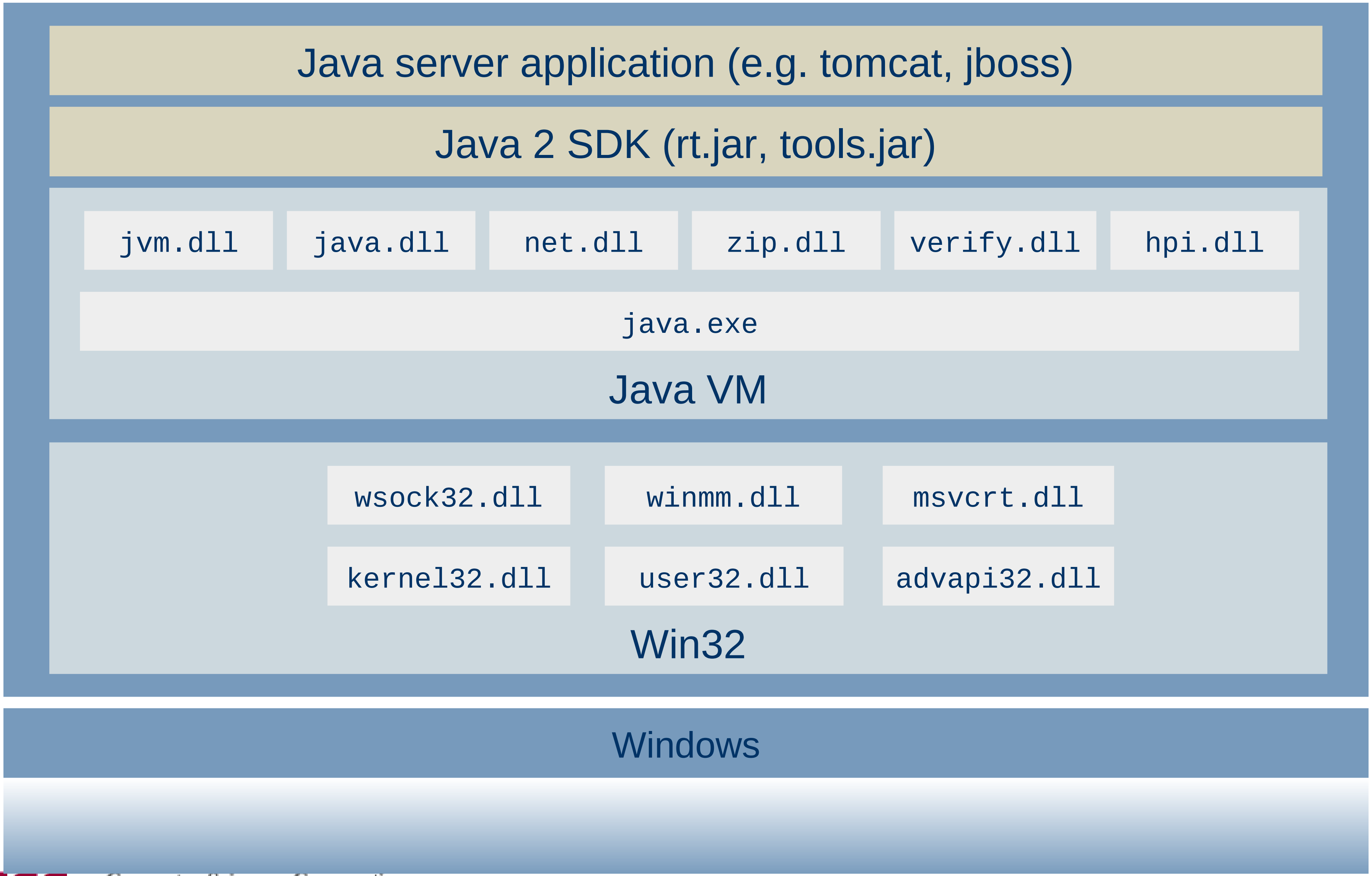
How to build a jbox...

Computer Sciences Corporation

Java and the operating system

- What is actually going on under the hood when you run a Java application?
- How is the JVM using the operating system?
- What features of the operating system are used by a Java server application?
- How do you build a JavaOS kernel?

Java VM on Windows



194 Windows API calls used

KERNEL32

- CloseHandle
- CreateEventA
- CreateFileA
- CreatePipe
- CreateProcessA
- CreateSemaphoreA
- DebugBreak
- DeleteFileA
- DisableThreadLibraryCalls
- DuplicateHandle
- EnterCriticalSection
- FindClose
- FindFirstFileA
- FindNextFileA
- FlushFileBuffers
- FormatMessageA
- FreeLibrary
- GetCurrentDirectoryA
- GetCurrentProcess
- GetCurrentThread
- GetCurrentThreadId
- GetEnvironmentVariableA
- GetExitCodeProcess
- GetFileAttributesA
- GetLastError
- GetLogicalDrives
- GetModuleFileNameA
- GetNumberOfConsoleInputEvents
- GetProcAddress
- GetStdHandle
- GetSystemDirectoryA
- GetSystemInfo
- GetSystemTime
- GetSystemTimeAsFileTime
- GetTempPathA
- GetThreadContext
- GetThreadLocale
- GetThreadPriority
- GetThreadTimes
- GetTimeZoneInformation
- GetVersionExA
- GetWindowsDirectoryA
- InitializeCriticalSection
- InterlockedDecrement
- InterlockedIncrement
- IsDBCSLeadByte
- LeaveCriticalSection
- LoadLibraryA
- PeekConsoleInputA
- PeekNamedPipe
- QueryPerformanceCounter
- QueryPerformanceFrequency
- ReleaseSemaphore
- RemoveDirectoryA
- ResetEvent
- ResumeThread
- SetConsoleCtrlHandler
- SetEndOfFile
- SetEvent
- SetFileAttributesA
- SetFilePointer
- SetFileTime
- SetHandleInformation
- SetThreadContext
- SetThreadPriority
- Sleep
- SuspendThread
- SystemTimeToFileTime
- TerminateProcess
- TlsAlloc
- TlsGetValue
- TlsSetValue
- VirtualAlloc
- VirtualFree
- VirtualQuery
- WaitForMultipleObjects
- WaitForSingleObject
- WideCharToMultiByte

USER32

- MessageBoxA

ADVAPI32

- GetUserNameA
- RegCloseKey
- RegEnumKeyExA
- RegOpenKeyExA
- RegQueryInfoKeyA
- RegQueryValueExA

WSOCK32

- __WSAFDIsSet
- accept
- bind
- closesocket
- connect
- gethostbyaddr
- gethostbyname
- gethostname
- getprotobyname
- getsockname
- getsockopt
- htonl
- htons
- ioctlsocket
- listen
- ntohl
- ntohs
- recv
- recvfrom
- select
- send
- sendto
- setsockopt
- shutdown
- socket
- WSACleanup
- WSAGetLastError
- WSAStartup

MSVCRT

- new
- delete
- __dllonexit
- __mb_cur_max
- _access
- _adjust_fdiv
- _assert
- _beginthreadex
- _CIfmod
- _close
- _control87
- _endthreadex
- _errno
- _except_handler3
- _finite
- _fstati64
- _ftol
- _fullpath
- _get_osfhandle
- _getdcwd
- _getdrive
- _initterm
- _iob
- _isctype
- _isnan
- _lseeki64
- _mkdir
- _onexit
- _open
- _open_osfhandle
- _pctype
- _purecall
- _read
- _setjmp3
- _setmode
- _stat
- _stati64
- _strdup
- _vsnprintf
- _write
- abort
- atof
- calloc
- exit
- fclose
- fflush
- fgets
- fopen
- fprintf
- fputc
- free
- getc
- getenv
- isalnum
- isspace
- longjmp
- malloc
- memmove
- printf
- putchar
- qsort
- raise
- realloc
- rename
- signal
- sprintf
- sscanf
- strchr
- strerror
- strncmp
- strncpy
- strrchr
- strstr
- strtol
- toupper
- vfprintf
- vsprintf

WINMM

- timeEndPeriod
- timeBeginPeriod
- timeGetTime

OS Services

- File system (`open, close, read, write, `...)
- TCP/IP network sockets (`listen, send, recv, `...)
- Virtual memory (`mmap, munmap, ...`)
- Threads (`beginthread, suspend, resume, setprio`, ...)
- Synchronization (`wait, mksem, mkevent`, ...)
- Time (`time, gettimeofday`, ...)
- DNS resolver (`gethostbyname`, ...)
- Heap allocator (`malloc, free, realloc`, ...)
- DLL modules (`load, resolve`, ...)
- Critical sections (`mkcs, enter, leave`, ...)
- Thread local storage (`tlsalloc, tlsset, tlsget`, ...)
- C runtime library (`strcpy, atol, sprintf`, ...)

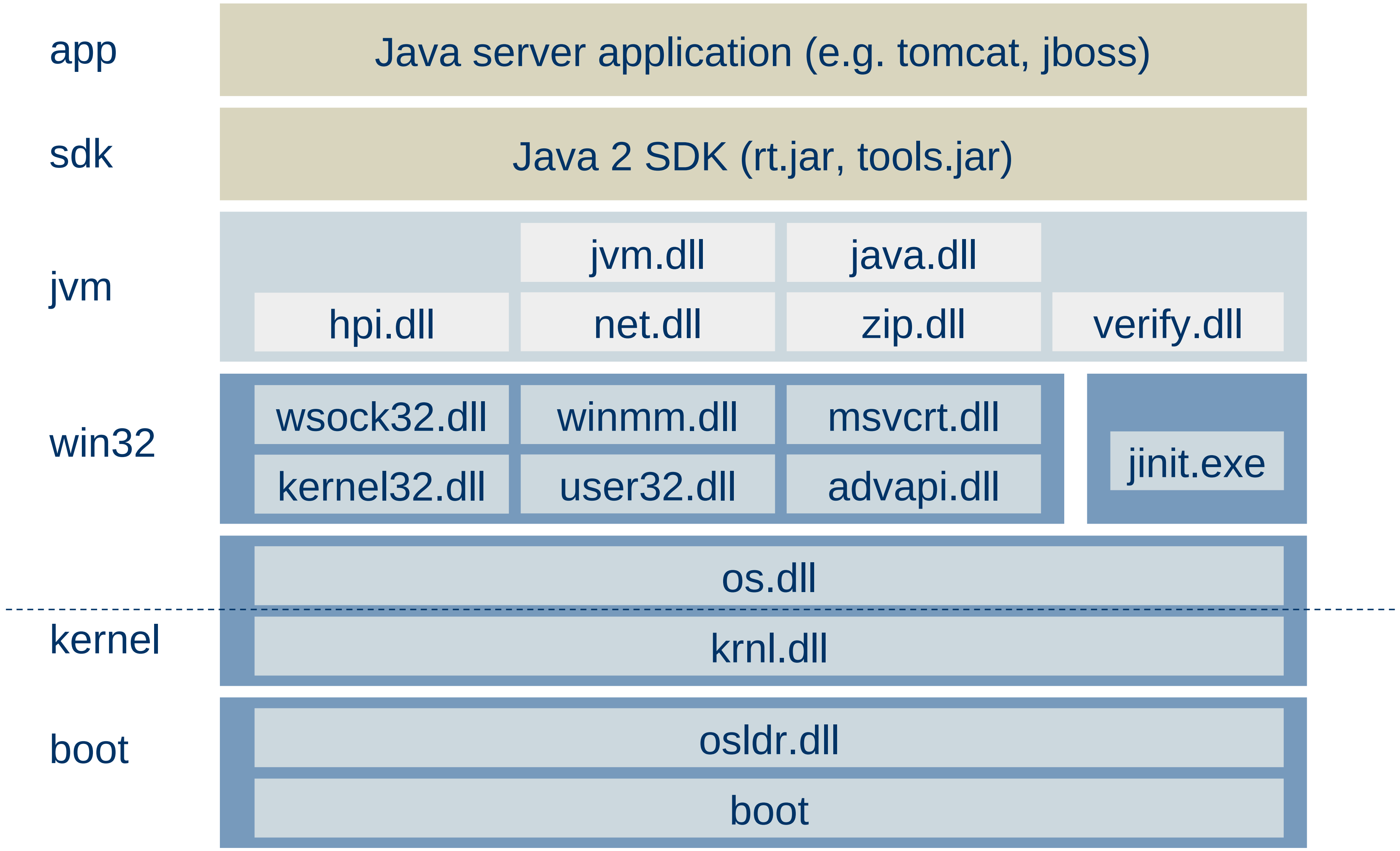
jbox.dk

sanos kernel

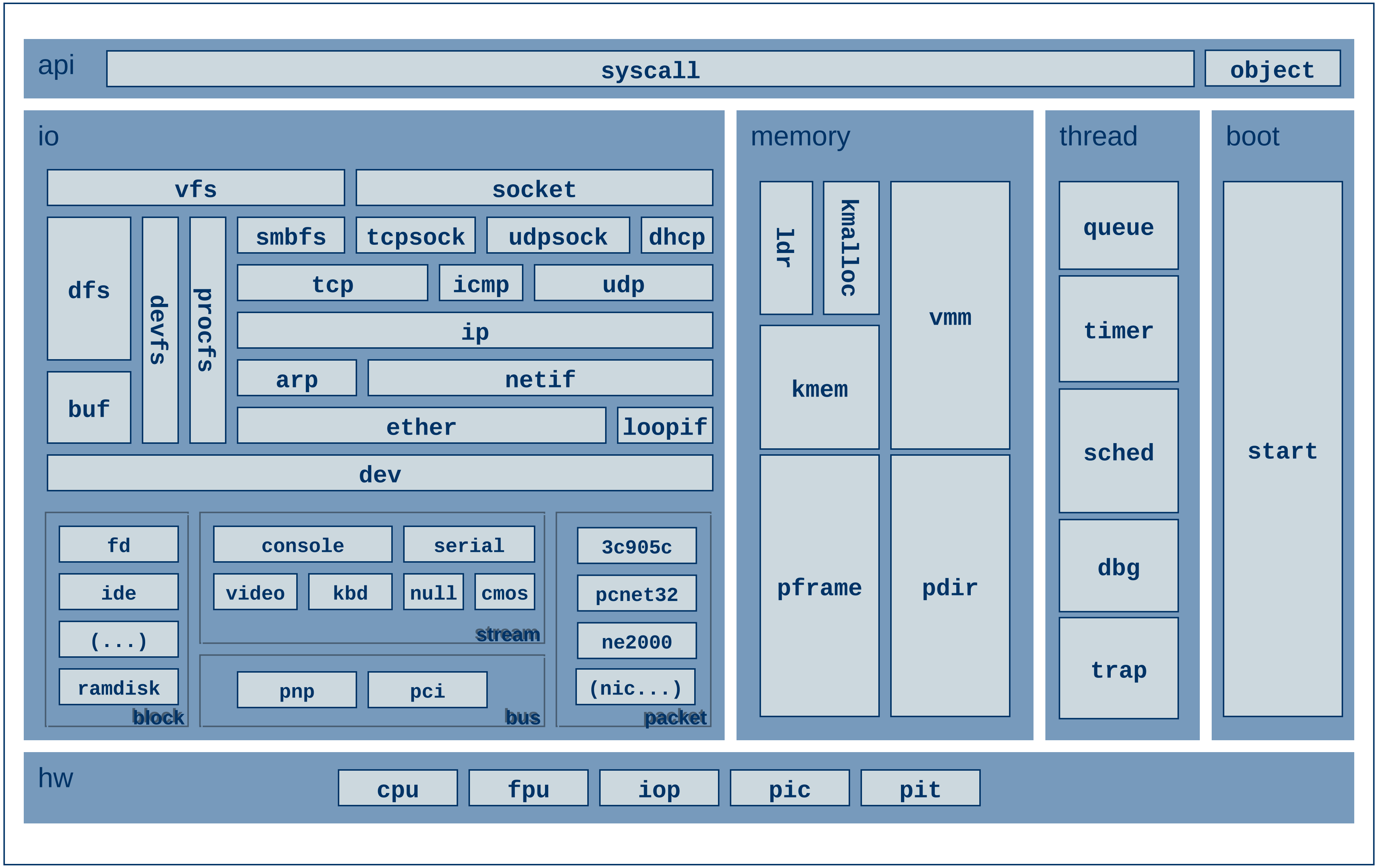
- A kernel for executing Java server applications on appliances
- Uses existing HotSpot VM for Windows
- Small, simple, fast but complete kernel
- Runs on standard PC hardware (IA-32)
- Developed using Microsoft Visual C
- Uses standard EXE/DLL executables

jbox.dk

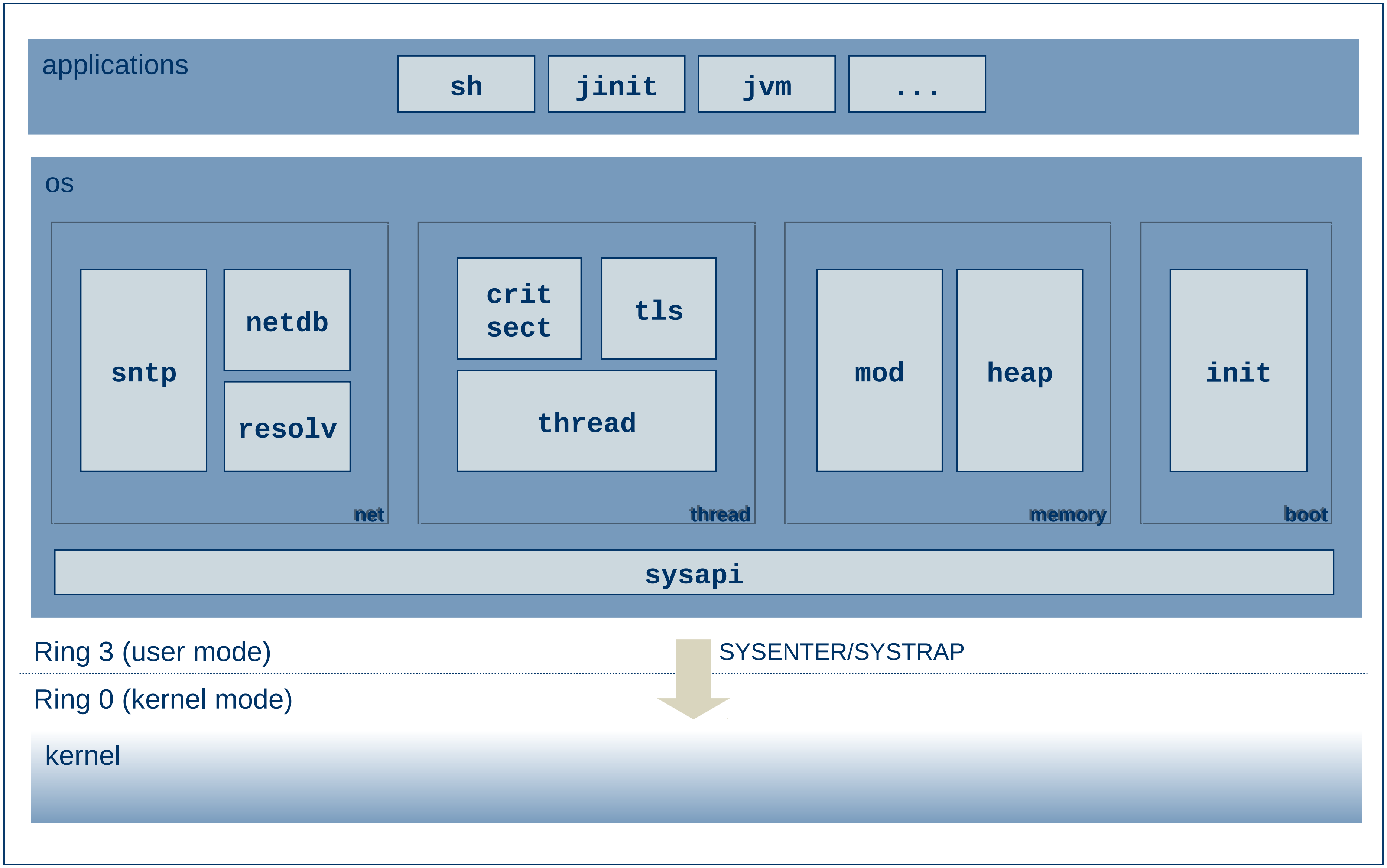
sanos architecture layers



Kernel Architecture



User Mode Components



jbox.dk

sanos API

file

- canonicalize
- chdir
- chsize
- close
- dup
- flush
- format
- fstat
- fstatfs
- futime
- getcwd
- getfsstat
- ioctl
- link
- lseek
- mkdir
- mount
- open
- opendir
- read
- readdir
- readv
- rename
- rmdir
- stat
- statfs
- tell
- umount
- unlink
- utime
- write
- writev

socket

- accept
- bind
- connect
- getpeername
- getsockname
- getsockopt
- listen
- recv
- recvfrom
- send
- sendto
- setsockopt
- shutdown
- socket

time

- clock
- gettimeofday
- settimeofday
- time

memory

- mlock
- mmap
- mprotect
- mremap
- munlock
- munmap

thread

- beginthread
- endthread
- epulse
- ereset
- eset
- getcontext
- getprio
- gettib
- gettid
- mkevent
- mksem
- resume
- self
- semrel
- setcontext
- setprio
- sleep
- suspend
- wait
- waitall
- waitany

system

- config
- dbgbreak
- exit
- loglevel
- panic
- peb
- syscall
- syslog

critsect

- csfree
- enter
- leave
- mkcs

tls

- tlsalloc
- tlsfree
- tlsget
- tlsset

heap

- calloc
- free
- mallinfo
- malloc
- realloc

module

- exec
- getmodpath
- getmodule
- load
- resolve
- unload

resolver

- dn_comp
- dn_expand
- res_mkquery
- res_query
- res_querydomain
- res_search
- res_send

netdb

- gethostbyaddr
- gethostbyname
- gethostname
- getprotobyname
- getprotobynumber
- getservbyname
- getservbyport
- inet_addr
- inet_ntoa

Files

sanos

```
/bin:
  krnl.dll          184.0 KB
  3c905c.sys         13.4 KB
  os.dll             35.4 KB
  sh.exe             22.0 KB
  jinit.exe           6.4 KB
  msvcrt.dll         18.4 KB
  kernel32.dll       18.4 KB
  user32.dll          2.4 KB
  advapi32.dll        4.4 KB
  wsock32.dll         5.0 KB
  winmm.dll           2.4 KB

/etc:
  krnl.ini            0.5 KB
  os.ini              1.7 KB
```

java

```
/usr/java/bin:
  hpi.dll             28.0 KB
  java.dll            88.0 KB
  net.dll             32.0 KB
  ioser12.dll         24.0 KB
  jpeg.dll           108.0 KB
  zip.dll             52.0 KB
  verify.dll          52.0 KB
  jcov.dll            40.0 KB
  hprof.dll           44.0 KB

/usr/java/bin/hotspot:
  jvm.dll            648.0 KB

/usr/java/lib:
  rt.jar           12811.4 KB
  i18n.jar          2576.8 KB
  tools.jar         4535.2 KB
  sunrsasign.jar      84.2 KB
  tzmappings           6.3 KB
  content-types.properties
                       5.3 KB

/usr/java/lib/security:
  cacerts              7.1 KB
  java.policy          2.1 KB
  java.security        3.8 KB
```

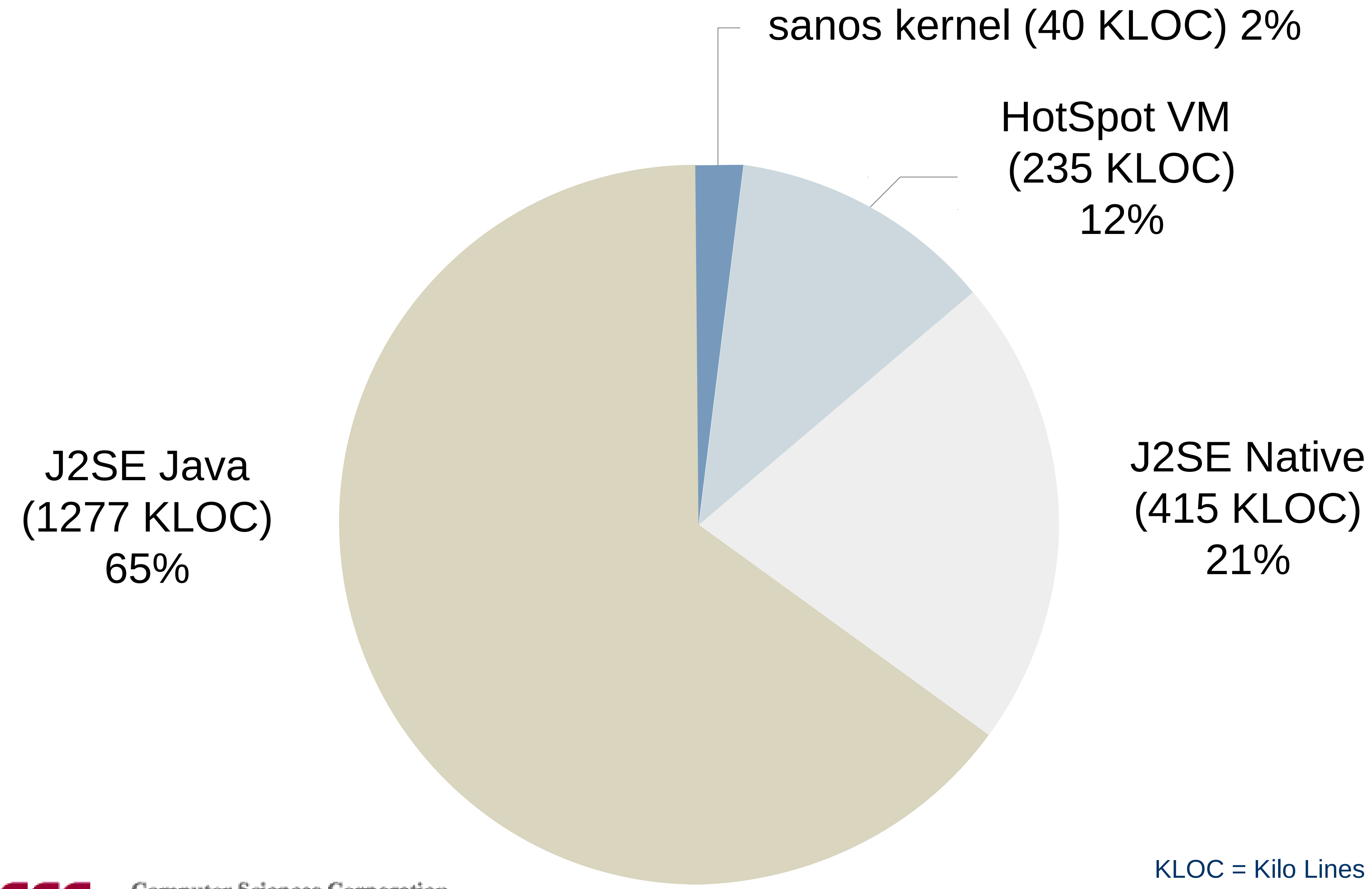
tomcat

```
/usr/tomcat/lib:
  webserver.jar   420.3 KB
  servlet.jar      39.8 KB
  jasper.jar      212.8 KB
  jaxp.jar          5.4 KB
  parser.jar      132.9 KB

/usr/tomcat/conf:
  server.xml        9.0 KB
  tomcat-users.xml 0.2 KB

/usr/tomcat/webapps:
  admin.war         6.3 KB
  examples.war    112.5 KB
  root.war        430.8 KB
  test.war         85.1 KB
```

Where is the code?



KLOC = Kilo Lines Of Code

OS Development on the Internet

- There are lots of information and code on the internet on OS topics:
  - Linux kernel code (www.kernel.org)
  - IA-32 Reference Manual (www.intel.com)
  - TCBs and u-kernels (Jochen Liedtke, http://i30www.ira.uka.de/teaching/coursedocuments/47/)
  - DNS Resolver (ISC BIND lwres, www.isc.org)
  - TCP/IP Stack (Adam Dunkels, www.sics.se/~adam/lwip/)
  - Heap Allocator (Doug Lea, http://gee.cs.oswego.edu/dl/html/malloc.html)
  - Bochs (bochs.sourceforge.net) and WMWare simulators (www.vmware.com)
  - IDE Disks (Hale Landis, www.ata-atapi.com)
  - ...

jbox.dk

Conclusion

Is Java an operating system ?

No, but if you add 2% to the code that is already there it can become an operating system!

Did we write our own operating system ?

No, we only made the kernel, SUN did the remaining 98%!

sanos has been released as open source (BSD license) and is available for download at www.jbox.dk