The K42 Research OS

IBM TJ Watson Research Center

M. Auslander, D. Da Silva, D. Edelsohn, O. Krieger, C. MacDonald,

B. Rosenburg, M. Ostrowski, B. Wisniewski, J. Xenidis

Partners : IBM Austin, Univ Toronto, CMU

http://www.research.ibm.com/k42

The K42 Research OS: Why

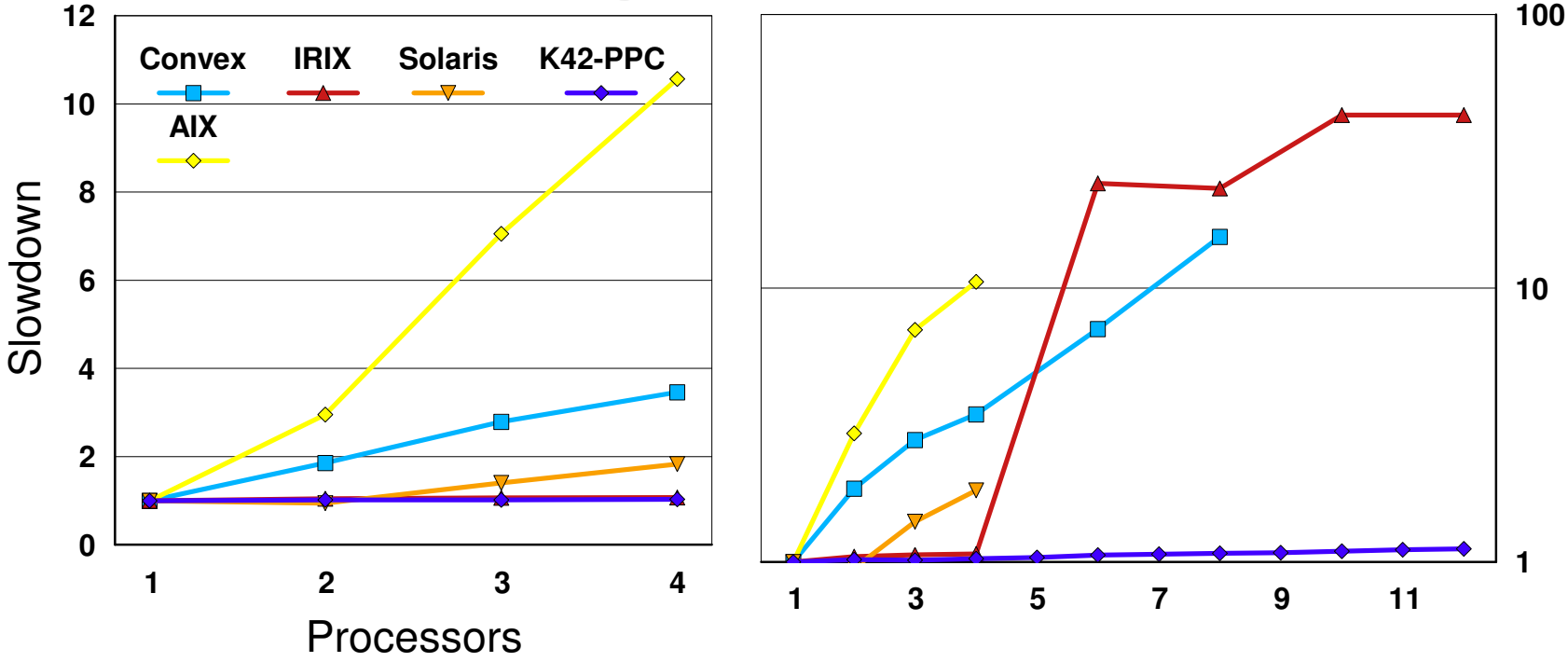
- core structures of current operating systems are based on twenty year old designs. Challanges:
  - scalability issues
  - maintainability, portability
  - new types of applications/users
- new 64-bit MP platforms provide opportunity for new operating system technology
- Linux gives us open interfaces to support for applications and devices

Project Goals

- Performance
  - scalability
  - flexibility/customizability
  - portability
- Other goals
  - Linux API/ABI
  - availability, fault containment
  - security
  - maintainability

Example of scalability issue:

Page faults



	System	OS	Pr
K42-PPC	IBM S70	Tornado	12
Solaris	450 Ultrasparc II	Solaris 2.5.1	4
AIX	IBM S70	AIX 4.2.0.0	4
IRIX	Origin 2000	IRIX 6.4	40
Convex	SPP-1600	SPP-UX 4.2	32

Key architectural features

- object-oriented structure for performance
  - every resource represented by a set of independent objects
  - no global locks, data structures, policies
  - objects "extra overhead" ? locality pays off
- obviously not enough for scalability

Key architectural features *(cont)*

Clustered Objects Model

- common techniques for scalability replication, distribution, partitioning are part of K42
- efficient implementation via per-processor translator table (one extra instruction)
- reps are created on demand by a mishandling object (~150 instructions)
- translation tables are large and sparse; kept on virtual memory (even in the kernel). Paging out can simply discard victim pages (mishandlers do backup)

- dynamic memory allocation
  - supports processor-local memory allocation
  - highly concurrent
  - maximize locality both in its internal design and in the memory returned
  - clusters (3 extra instructions on critical path of free)
  - user level allocation without disabling interrupts
- locking:
  - efficient spin-then-block
  - semi-automatic garbage collection for clustered objects avoids the existence guarantee problem

Inter-process communication

- Protected Procedure Call (PPC)
  - client requests are always serviced on their local processor (handoff scheduling)
  - as many threads of control as clients
  - client specific state maintained in the server can be hold local to the client
  - stubs automatically generated
  - security check
  - common case call and return requires 372 instructions

Hot-Swapping Live Component Software

- Replacement a component while it is in use.
  - Automatically or manually. System wide or by instance.
- Switch to better implementation and/or policies based on resource use.
- Dynamically update components for high availability systems.

- Basic algorithm
  - instantiate the replacement component
  - establish a quiescent state (temporarily idle)
  - transfer state
  - swap, adjusting all references
  - deallocate old component
- generation counter mechanism
- mediator object; 3 phases
- cost of first mediated call after swap initiation:1815
- performance evaluation: counter/grep/SDET

Performance Monitoring

- Two mechanisms where tracing results can be obtained.
  - SimOS Trace Facility (simulator only)
    - Annotations: Tcl scripts that run on events
      - Complete, non-intrusive access to machine state.
  - Kmon (simulator and hardware)
    - K42 specific tool allows easy instrumentation of event data.
    - Java based data viewer and analyzer.
    - Data available to K42 for self analysis.
    - Visualization crucial for understanding complex system behavior.

Multilevel Processor Scheduling

- What is it?
  - Divide responsibility for scheduling between user-level and kernel-level code.
- Why?
  - By putting part of the scheduler in the user-level code, kernel interactions are reduced.
  - Greater scheduling flexibility at user level.
- more fully preemptable and pageable than most operating systems

Performance

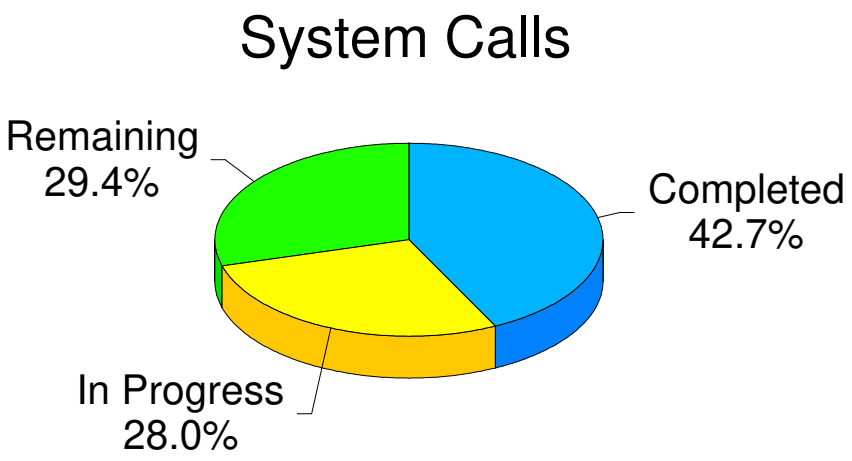
- microbenchmarks: under extreme load, uni-processor competitive performance
- SPEC SDM
  - Software Development Environment Workload (SDET)
- SPLASH-2
  - Positive preliminary results on 2, 4 and 8 way.
  - In Progress
    - Gather results (if any) specific to Gang Scheduling

Status

- platforms: 64-bit PowerPC SMP and 64-bit MIPS NUMA system (being ported to AMD Sledgehammer)
- micro-benchmark results demonstrate:
  - uniprocessor performance competitive with best commercial and research systems
  - small-scale MP performance better (x10 on 4 processors) than other systems
  - scalability better (x100 on 16 processors) than other systems

Linux Functionality

(System Calls [196])



- Completed
  - File access
  - Process Hierarchy
  - Sockets and pipes
- In Progress
  - Process control
  - Address Space
  - Signals and tty/pty

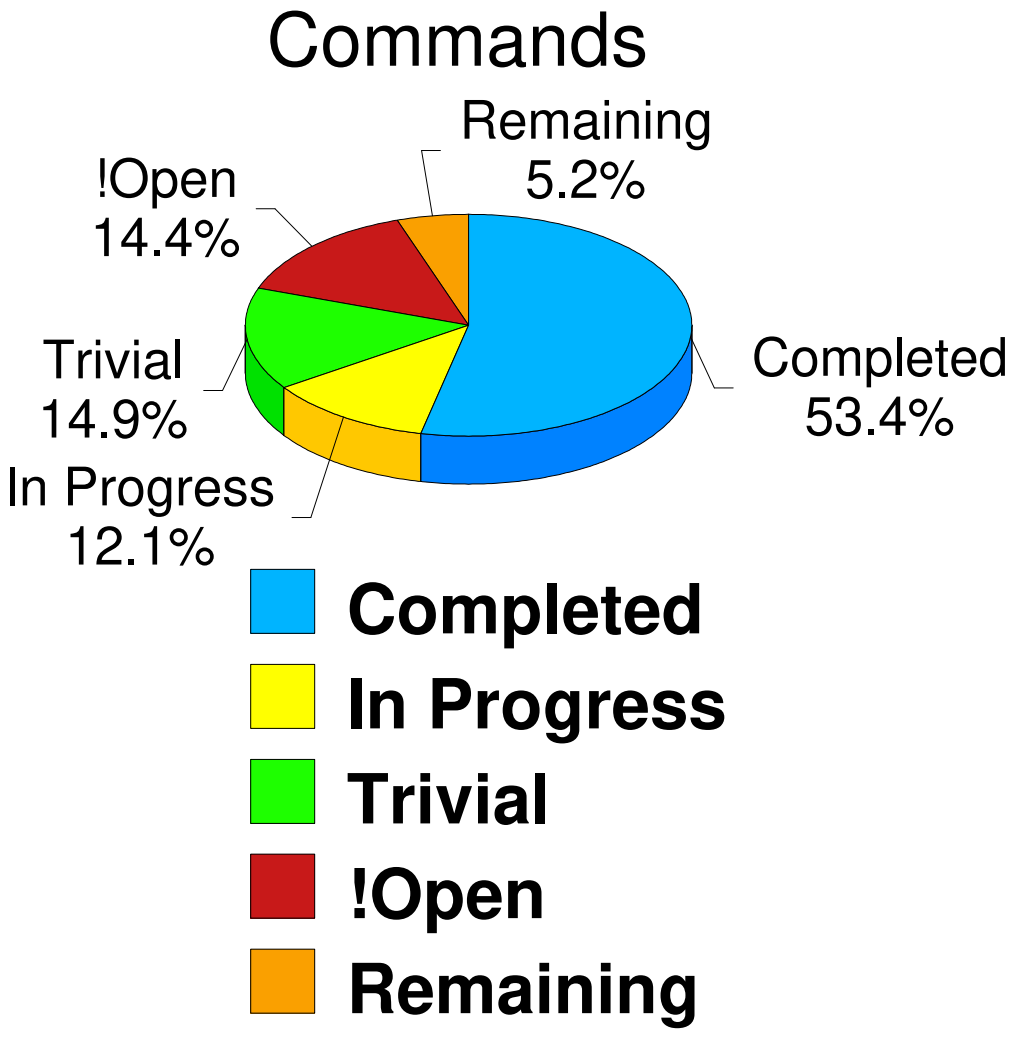
Remaining

- Accounting (sar)
- Kernel module management
- Process scheduling (nice)
- IPC (shmat, AF_UNIX, mqueues)

Completed
In Progress
Remaining

Linux Functionality

(Unix98 Commands [175])



- In Progress
  - Tool Chains (Linker, Compiler)
  - Full Screen (curses apps)
  - Kernel Info (ipcs, ps, logger)
- Completed
  - Shells
  - File / Dir. Utilities
  - Text Manipulation

Trivial

- Sequential file programs (tar, patch, calendar, etc.)

!Open Source

- SYSV only (SCCS, pack)

Remaining

- UUCP

Linux Functionality

(Drivers & Daemons)

- Native to K42
  - Done
    - NFS
    - ToyFS
    - DevFS
    - Rlogind
  - In Progress
    - Unix98 PTY's
    - HFS
- K42 Linux Driver Layer
  - Done
    - TCP/IP Stack
    - Ethernet
  - In Progress
    - SCSI (~complete)
  - Future
    - Linux Vnode layer

Problems

- 64-bit platforms have not taken off yet, hard to get cheap systems to collaborators
- PowerPC has been a huge time sink, we now maintain gcc, SimOS, and have had to supervise toolchain...
- open source and linux efforts consume time
  - FSF: Gcc (C/C++),PPC32/64 XCOFF and ELF targets; Binutils:
  - Consultation on PPC32/64: XCOFF with Cygnus, LF with SWOX

Cool technology

- user-level threading model: threads, IPC model, locality preserving, load balancing, ...
- scheduling infrastructure: real-time, background, gang scheduling...
- locking architecture/type safe memory
- clustered objects
- customizability: building-blocks, dynamic object-switching, open-source model
- memory allocation
- Linux compatibility (strategy, API, ABI, 32-bits)
- trace/performance monitoring infrastructure
- portability model
- fault containment...